KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering

Exploiting Appearance-based Representations for Recognition

Amir Ghodrati

Supervisor:
Prof. dr. ir. T. Tuytelaars

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

December 2016

Exploiting Appearance-based Representations for Recognition

Amir GHODRATI

Examination committee:
Prof. dr. ir. J. Vandewalle, chair
Prof. dr. ir. T. Tuytelaars, supervisor
Prof. dr. ir. L. Van Eycken
Prof. dr. M.-F. Moens
Prof. dr. ir. T. Goedemé
Dr. J. Verbeek
(INRIA Rhône-Alpes)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

December 2016

Uitgegeven in eigen beheer, Amir Ghodrati, Kasteelpark Arenberg 10 box 2441, B-3001 Heverlee (Belgium)

Acknowledgements

After more than 10000 hours of working and around $2222km$ of walking and $1000km$ of biking back and forth from my place to the work, it is time to say my last words of this journey. Working toward a PhD is not something that can be done without help of many people. During my journey toward PhD, I had the good fortune of meeting many people who supported me in various ways. First of all, I want to express my gratitude to my wonderful supervisor, Prof. Tinne Tuytelaars. She is the best supervisor I could ever have, possessing broad and deep scientific knowledge, while always being kind and sympathetic. The freedom she gave me in my research is also very much appreciated. I would like to thank all members of my supervisory committee and examination committee for their useful remarks throughout my Ph.D.

I have collaborated a lot during my PhD and I am happy that many people share their ideas with me. I want to give special thanks to Dr. Marco Pedersoli, with whom I collaborated for a significant part of my PhD. I very much enjoyed from discussions and collaborations we had. I thank Xu Jia, Dr. Basura Fernando, Dr. Efstratios Gavves, Dr. Jose Oramas, Ali Diba, Roeland De Geest, Prof. Luc Van Gool and Prof. Cees Snoek for all interesting collaborations we did together.

I thank all friends and colleagues for creating such wonderful environment. I thank Vincent, Stam, Kostas, Jay, Rahaf, Wacha, Amal, Maxim, Gina, Klaas, Xuanli, Tom, Yu-Hui, Bert, Matthew, Jan-Pieter, Vivek, Davy, Tomas, Rosalia, Rodrigo, Marcus, Angelo, Hakan and Chris. I am also grateful to Paul Konijn for his great support. I also would like to give special thanks to all my friends[1] for all the awesome time we shared together. Guys, you are the best gift for me. I want to give special thanks to Asefeh. I appreciate how positive and also

[1] Hossein, Amir×2, Hana, Mahoor & Myrna, Mamad×2, Shayan & Julie, Ali×4, Alireza, Homayoon, Nima×2, Marco & Bahar, Neda, Omid, Ariana, Celia, Leidy, Xiaohan, Sheida, Mattia, Farzad & Liesbeth, Keivan & Lisa, Milad, Amir Hossein & Neda, Babak, Hossein & Sahar, Morshed & Mona, Amir Hossein, Ehsan and Payam.

tolerative she was during my PhD.

I thank the financial support from the DBOF PhD scholarship, FWO project and the FP7 ERC grant 240530 COGNIMUND.

Last and most importantly, I am very thankful to my parents, my sister, Mehdi and Kasra for their unconditional support. Their love and actions have been invaluable to me. This thesis is dedicated to my deceased father who would like to see my graduation.

Abstract

It is widely accepted that the success of vision algorithms depends to a large extent on the chosen image or video representation. Different representations can capture different semantic factors of the data and bring robustness to various factors such as image noise, clutter, blur, etc. In this thesis, we concentrate our efforts to find state-of-the-art appearance-based image and video representations for various computer vision tasks namely action recognition, viewpoint estimation, object proposal generation, and image generation.

We start the thesis by making use of bag-of-words, one of the most commonly used representations in object and video recognition, for the task of action recognition. We propose and evaluate different ways to integrate motion segmentation and action recognition. We obtain state-of-the-art results on two benchmarks, and show that these two tasks are interdependent and an iterative optimization of the two gives best results.

Afterwards, we exploit the next generation of representations based on Fisher encoding to estimate viewpoint of objects. We show how solely using such 2D representations, if properly tuned, enables us to effectively bypass 3D models and obtain promising results in estimating viewpoint of faces, cars and general objects.

Next, we make use of data-driven, learning-based representations for generating a set of object proposals in an image. Particularly, we propose an efficient coarse to fine cascade on multiple layers of a deep convolutional neural network that acts strongly on object and action locations. Our method in most of the cases is comparable or better than state-of-the-art approaches in terms of both accuracy and computation.

Finally, observing the power of the deep learning paradigm, we tackle the problem of image generation to evaluate to what extent a representation is inversely convertible to an image. To this end, we define the new problem of generating modified images using a deep encoder-decoder architecture. We

obtain good qualitative and quantitative results on a face dataset on three sub-tasks, that is, rotating faces, changing illumination and image inpainting.

Beknopte samenvatting

Het succes van computervisiealgoritmes hangt voor een groot deel af van de gekozen beeld- of videorepresentatie. Verschillende representaties kunnen verschillende semantische aspecten van de data vatten en geven een robuustheid jegens verscheidene factoren zoals beeldruis, clutter, onscherpte, etc. In deze thesis concentreren we ons op het vinden van state-of-the-art verschijningsgebaseerde beeld- en videorepresentaties voor verscheidene computervisietaken zoals actieherkenning, gezichtspuntschatting, generatie van voorwerpshypotheses en beeldgeneratie.

We starten de thesis met het gebruiken van *bag-of-words*, een van de meest gebruikte representaties in voorwerps- en videoherkenning, voor de taak van actieherkenning. We introduceren en evalueren een methode om, op verschillende manieren, bewegingssegmentatie en actieherkenning te integreren. We behalen state-of-the-art resultaten op twee criteria en tonen aan dat deze twee taken van elkaar afhankelijk zijn en dat een iteratieve optimalisatie van de twee de beste resultaten geeft.

Vervolgens gebruiken we de volgende generatie van representaties gebaseerd op Fisher encodering om gezichtspunten van voorwerpen te schatten. We laten zien hoe we, enkel gebruikmakend van zulke 2D representaties en met de juiste afstelling, 3D modellen kunnen vermijden en veelbelovende resultaten bekomen in het schatten van het gezichtspunt van gezichten, auto's en algemene voorwerpen.

Daarna schakelen we over op datagedreven, aangeleerde representaties om een set van voorwerpshypothesen in een afbeelding te genereren. Hiervoor stellen we een efficiënte grof-naar-fijn cascade voor die toegepast wordt op meerdere lagen van een diep convolutionair neuraal netwerk dat getrained werd voor voorwerps- of actie-localizatie. Onze methode is vergelijkbaar of beter dan state-of-the-art methodes op vlak van accuraatheid en rekentijd.

Tot slot, de kracht van het *deep learning*-paradigma indachtig, behandelen

we het probleem van beeldgeneratie om te evalueren tot op welke hoogte vanuit een dergelijke representatie een afbeelding gereconstrueerd kan worden. Hiervoor definiëren we een nieuw probleem dat gewijzigde afbeeldingen genereert gebruik makend van een diepe encoder-decoder-architectuur. We behalen goede kwalitatieve en kwantitatieve resultaten op een gezichtsdataset voor drie deeltaken, namelijk roterende gezichten, veranderende belichting en beeldinvulling.

Abbreviations

AVP	Average Viewpoint Precision
BoW	Bag of Visual Words
CAD	Computer-Aided Design
CAE	Convolutional Auto Encoder
CNN	Convolutional Neural Network
DeCAF	Deep Convolutional Activation Features
DPM	Deformable Part Model
FV	Fisher Vector
GAN	Generative Adversarial network
GMM	Gaussian Mixture Model
HOF	Histogram of Optical Flow
HOG	Histogram of Oriented Gradients
IoU	Intersection Over Union
LLC	Locally-constrained Linear Coding
MAE	Median Angular Error
MBH	Motion Boundary Histogram
NMS	Non Maximum Suppression
ReLU	Rectifier Linear Unit
SIFT	Scale-invariant Feature Transform
SP	Spatial Pyramid
SVM	Support Vector Machine
VQ	Vector Quantization

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cameras are appearing everywhere: cameras built into mobile phones or wearable devices like glasses; surveillance cameras around the campus; cameras taking medical images such as ultrasound and X-ray; speed cameras; cameras on satellites; the list keeps expanding. With such advancement in image capturing devices, the amount of visual data has grown significantly both quantitatively and qualitatively. At present many of these images are just looked at by people. However the processing of images from all these cameras can be automated allowing us to do things never considered before: smart phone cameras can be used to capture images of signs and automatically translate them; surveillance cameras to identify thieves and alert security; medical cameras to diagnose conditions more reliably than the best expert; speed cameras to read the plate number of cars with speed above the speed-limit; and someday cameras on robots allowing them to interact with the world just like us!

Computer vision is the field of making computers or other machines able to see. However, seeing is more than the process of recording light in a form that can be played back, like taking a photo. The ultimate goal of computer vision is to computationally model human vision using computer algorithms at both low-level and high-level. In *low-level computer vision* an image is processed for extracting information like edge, corner, or optical flow. *High-level computer vision* aims to interpret the information provided by the low-level information in form of tasks like object recognition or motion analysis. The goal of high-level vision is to build a system that is able to interpret and understand the environment same as a human is able to. Generally, the output of such system is a description or an interpretation or some quantitative measurements of the image. In this thesis, we focus mainly on high-level vision tasks.

Considering high-level vision, in many respects it is an AI-complete problem: building general-purpose vision machines would require solutions to most of the general goals of artificial intelligence. This is consistent with scientific findings that more than half of our brains are involved in visual perception [Bowan and Optometrist, 2012]. Tackling a general-purpose vision task requires finding ways of building flexible and robust *visual representations* whose manipulation allows the machine to interact intelligently with the world.

In general, there are several steps being followed to build a computer vision system. The first step is data acquisition. The data that the system gets, is an image. Images are made up of pixels, each consists of 3 values for representing red, green and blue colors and 2 values for representing the location of that pixel in the image plane. Therefore, there are 5 variables in total associated with each pixel in an image: 2 for location and 3 for color. This means that in a 1024×768 image, there are $786,432 \times 5 = 3,932,160$ data items! Considering such huge amounts of data embedded in an image, the next step in building vision systems as a necessity is to design proper visual representations. The main challenge in this step is to find a representation, as small as possible, that keeps key visual elements and information from the image. Finally in the last step, depending on the target task, different machine learning algorithms are used to process the output of the previous step. For example in the task of assigning an input image one label from a fixed set of categories, a machine learning procedure called *classification* can be used.

The success of vision algorithms depends to a large extent on the chosen representation. Indeed, different representations can capture different semantic factors of the data. We believe that computer vision is at a stage where designing a good representation is a key to progress. There are huge amounts of work dedicated for finding effective image representations during last couple of decades. These efforts can be categorized into two main approaches. The first approach is based on designing hand-crafted representations. This type of representation is able to incorporate directly human ingenuity and prior knowledge. It enjoys the flexibility and computational efficiency, and does not rely on large sets of samples for training. However, these hand-crafted representations often rely on expert knowledge. Also, they normally do not generalize well. This motivates researches to go toward the second main approach which is based on learning the representation. Representation learning can directly learn data representations from raw training samples in an end-to-end fashion and detect data-driven features for a specific task. In contrast to feature engineering, representation learning can exploit the structure underlying the input data automatically and extract and organize the discriminative information from it. This leads to a representation with higher generalization ability. But this requires a large set of training images (sometimes called big data). Time-wise, these two main

approaches can be distinguished from the time deep learning techniques in combination with big data re-emerged. Before the era of recent deep learning, engineering the representation was the most popular and powerful approach. Recently, after re-emerging deep learning, representation learning has drawn increasing interest in visual recognition. Deep learning enables learning data-driven, highly representative, layered hierarchical image representations from sufficient training data. In this thesis we exploit both types of representations to tackle the task in hand.

In another representation categorization, we can roughly group different representations into two categories namely appearance-based representation and geometry-based representation. Geometry-based approaches are object-centered representations that usually store an explicit three-dimensional description of the object in some world coordinate frame. This representation relies on shape primitives like simple polyhedral models, generalized cylinders or curve sets with various invariant properties as basis for object representation. Since these primitive shapes are explicitly defined by numerical parameters, as more and more parameters are used, it takes fewer and fewer primitives to describe complicated objects. On the other hand, appearance-based approaches are viewer-centered representations that usually store intrinsic object and scene properties such as reflectance, color or texture information. In a hierarchical appearance-based representation, high-level abstract information and concepts like "wheels of a car" can be induced from lower-level appearance information. In this representation, an image is represented by a vector in high-dimensional feature space. It is well-known that appearance is often a more powerful discriminator rather than geometry in the recognition task mainly because of using a high level of abstraction in geometry-based representations. In this thesis, we focus on appearance-based representations.

1.1 Tasks of Interest

There are a variety of problems addressed in the computer vision literature, e.g. object detection, image/action classification, image/motion segmentation, single/multiple view 3D reconstruction, visual tracking, human pose estimation, image retrieval, image generation, etc. In this thesis we focus our attention on four problems *i.e.* action classification (Fig 1.1), viewpoint estimation (Fig 1.2), object detection (Fig 1.3) and image generation (Fig 1.4). The first three tasks are a subset of a higher level concept known as *recognition* while the last task is known as *generation*. Observing the power of state-of-the-art representations, in chapter 6, we move from recognition to the harder task of generation.

Figure 1.1: The task of action recognition and motion segmentation: **Left)** A frame sampled from an action clip. The aim is to classify this action as "skating". **Right)** segmenting each feature point (specified by dots) to action-related foreground (red) and action-unrelated background (yellow)

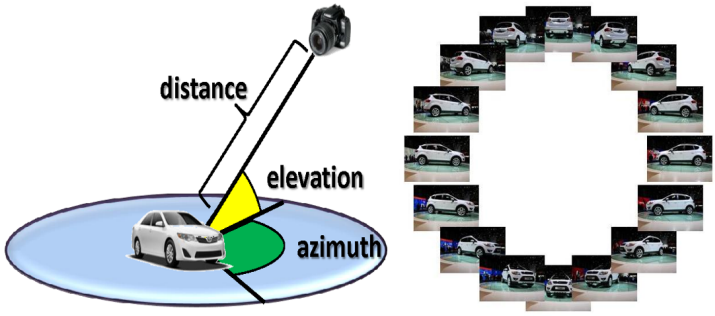

Figure 1.2: The task of viewpoint estimation: **Left)** Angles of the object in the 3D coordinate. we aim to estimate azimuth angle in the task of viewpoint estimation. **Right)** An object of interest in different viewpoints.

The intent of action classification is to categorize an action clip into one of several predefined classes. We do classification while leveraging the feedback from a segmentation process (*i.e.* partitioning the video to action-related foreground and action-unrelated background). We analyze our approach on two action datasets with different number of categories.

Viewpoint estimation aims at predicting a discrete or continuous azimuth angle (*i.e.* viewpoint or pose) of an object. In this thesis we perform our analysis on cars, faces and general objects.

In the task of object detection, we aim to first generate "proposals" *i.e.* a

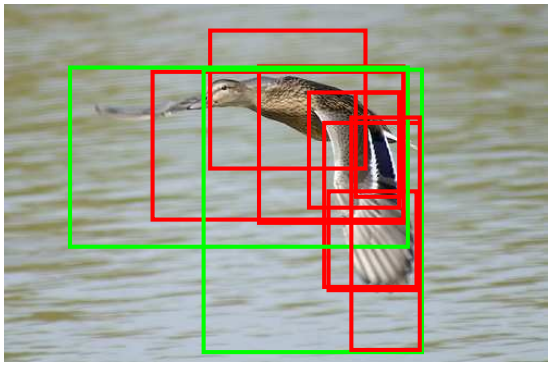

Figure 1.3: The task of generating proposals: generating a set of object hypotheses in an image. Proposals that their overlap with ground-truth object bounding box is more than 0.5 are shown with green color.

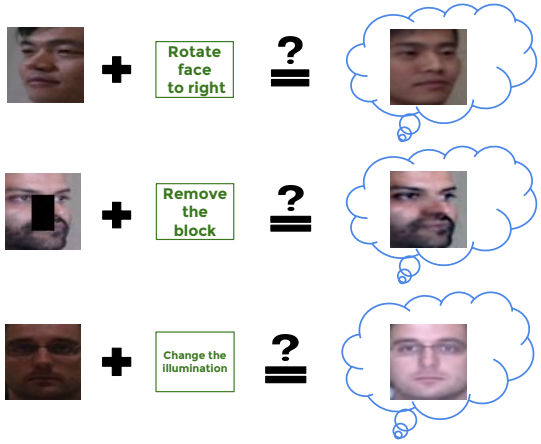

Figure 1.4: The task of image generation for three different sub-tasks: (**Left**) rotating faces, (**Middle**) image inpainting and (**Right**) changing the illumination.

reduced set of window candidates in an image that most likely contain an object (see Fig 1.3). Then we assess each of these candidates to detect and localize an object of interest. The PASCAL VOC dataset [Everingham et al., 2010] consisting of 20 object classes is used mostly for this task, as we do.

Finally, we tackle the task of image generation. We define a special image

generation task that aims at automatically editing an image by altering one of its attributes. More specifically, given an image of a certain class (e.g. a human face), the method should generate a new image as similar as possible to the given one, but with an altered visual attribute (e.g. the same face with a new pose or a different illumination). We evaluate the proposed method on MultiPIE face dataset for three sub-tasks, that is, rotating faces, changing illumination and image inpainting (see Fig 1.4).

There is a variety of applications that can benefit from our findings, e.g. detecting a face, estimating its viewpoint angle, rotating the face or changing its illumination.

1.2 Goal of the Thesis

The overall goal of this thesis is to investigate ways of exploiting state-of-the-art appearance-based image representations for various computer vision tasks, namely action recognition, viewpoint estimation, object detection and image generation. To reach this goal, in this thesis we use both hand-crafted representations as well as learning-based representations. More specifically, we aim to:

- Exploit properties of hand-crafted representations for the task of action recognition. Within the field of object recognition and image understanding, it is widely accepted that, for good results, segmentation and recognition should go hand in hand and be solved simultaneously. Therefore, our goal is to integrate, at different levels, segmentation and recognition using representations that enable us to formulate both of two tasks in one framework.

- Investigate the performance of modern representations for the task of viewpoint estimation. Recent top performing methods for viewpoint estimation make use of 3D information to build a 3D representation of the class. We want to study the impact of using only 2D information based on hand-crafted or learning-based representations on the performance of pose estimation.

- Exploit the hierarchy of a coarse-to-fine, learning-based representation in order to produce fast proposals for object detection and action localization. Our goal is to build an inverse cascade that, going backward from the high-level to the low-level layers of a deep neural network, selects the most promising locations for object detection.

- Learn a representation which best describes an unseen desired image. More specifically, given an input image and a desired change in a visual component of the image, our goal is to build a representation for the unseen, desired image using an encoder-decoder pipeline. Such representation should be as close as possible to the representation of input image yet embed information of the desired changes.

1.3 Motivation and research question

Data, representation and modeling are three important factors in every computer vision task. It is well-known that the scale of data is a key player to push computer vision to new frontiers. In order to learn anything useful,there is a need for "big data". Fortunately recently there is an increasingly rich amount of visual data available in our daily life (*e.g.* from our smart phones or image host services like Instagram and Flickr). Such emergence of "big data" has brought a paradigm shift throughout computer science. Same as data, good representation is also a key to progress. Different representations are designed typically to tackle different tasks in computer vision. It is known that the choice of representation depends on the task and also data. As a result, choosing a proper representation has a big impact on the final results. In addition to a good representation, a good learning model is also crucial in order to discriminate desired appearance variations for the specific task in hand. For example for the task of viewpoint estimation, variation in pose should be retained during learning while variations in color and object instances should be discarded.

Considering these points, the objective of this thesis is to exploit state-of-the-art appearance-based image representations for some computer vision challenges mentioned in section 1.1 and identify their strengths for those tasks. To this end, the thesis addresses the research question:

what makes a representation suitable for a specific vision task?

In this thesis, we use different representations for different tasks. For clarity of presentation, we split the main question into four questions to address specifically:

1. To what extent does disentangling foreground and background representations affect classification? Is quality of disentangling important for the final task of classification?
2. Are 2D-based representations enough for estimating viewpoint of an object?

3. How information from a hierarchical representation can be exploited for locating objects?

4. To what extent is a representation inversely convertible to an image?

The journey aimed at answering these research questions resulted in the following contributions.

1.4 Overview and contributions of the Thesis

The work covered in this thesis has been published in several papers. The contributions of the papers are presented in chapter 3 to 6. As a whole, the contents of these papers address the research questions introduced earlier with each chapter having specific contributions.

In chapter 2 we introduce some of the background topics and fundamental principles that are useful to understand later chapters. The objective of this chapter is to lay the foundations for the rest of the thesis.

In chapter 3 of this thesis we show that splitting the representation of an action into action-related foreground and action-unrelated foreground can be beneficial for the task of action recognition. The contributions of this part are **i)** to propose and evaluate several ways to integrate and combine motion segmentation and action recognition: 1) recognition using a standard, bottom-up segmentation, 2) using a top-down segmentation geared towards actions, 3) using a segmentation based on inter-video similarities (co-segmentation), and 4) tight integration of recognition and segmentation via iterative learning; **ii)** to show that the two task of segmentation and recognition are interdependent and a good segmentation is actually very important for recognition. *This work is published in the IEEE Winter Conference on Applications of Computer Vision (WACV) in 2014.*

Chapter 4 presents a study of different representation techniques for viewpoint estimation on four well-known and challenging datasets. The contributions of this part are **i)** to show comparable performance can be obtained just using 2D information without any 3D information that generally top performing methods use; **ii)** to consider viewpoint estimation as a 1-vs-all classification problem on the previously detected object bounding box; **iii)** to compare several features and parameter configurations and show that the modern representations based on Fisher encoding and convolutional neural network based features together with a neighbor viewpoints suppression strategy on the training data lead to promising performance on viewpoint estimation. *This work is published in the British Machine Vision Conference (BMVC) in 2014.*

Chapter 5 addresses the problem of object and action proposals. In this chapter we show how to leverage the hierarchical nature of deep convolutional neural networks to propose a new method for generating object and action proposals in images and videos. The contributions of this part are **i)** the insight that deeper layers of a neural network are highly informative for finding the objects and early layers are more powerful for localizing the object boundaries accurately; **ii)** to build an inverse cascade that, going backward from the later to the earlier convolutional layers of the CNN, selects the most promising locations and refines them in a coarse-to-fine manner; **iii)** to show that the method is efficient and also accurate: our method outperforms most of the previously proposed object proposal and action proposal approaches and, when plugged into a CNN-based object detector, produce state-of-the-art detection performance. *This work is published in IEEE International Conference on Computer Vision (ICCV) in 2015 and an extended article is under review for the International Journal of Computer Vision (IJCV).*

In chapter 6 we propose a method that aims at automatically editing an image by altering its attributes. More specifically, the contributions of this part are **i)** definition of a new problem where the goal is to generate images as similar as possible to a source image yet with one attribute changed; **ii)** a solution that follows an encoder-decoder pipeline, where the desired attribute modification is first encoded then integrated at feature map level; **iii)** the insight that the result can be refined by adding another convolutional encoder-decoder model; and **iv)** good qualitative and quantitative results on MultiPIE dataset on three sub-tasks, that is, rotating faces, changing illumination and image inpainting. *This work is published in British Machine Vision Conference (BMVC) in 2016.*

Chapter 7 concludes the thesis. This chapter begins by a general conclusion followed by revisiting the research questions devised in 1.3. Then we discuss the results we obtained and suggest directions for future research.

Chapter 2

Background

In this chapter we introduce some of the background topics on top of which the methods in other chapters are built. We start by introducing two well-known, traditional image representations namely Bag of visual Words (BoW) and Locally-constrained Linear Coding (LLC), which we will use in chapter 3 for action representation.

Next, in section 2.3 we introduce a more recent image representation method, Fisher encoding, which is used in Chapter 4 for viewpoint representation.

Afterward we introduce spatial pyramids in section 2.4, a type of pooling which can be used on top of any image representation (like BoW or Fisher encoding). We use spatial pyramids in Chapter 4 and 5.

In section 2.5 we briefly explain convolutional neural networks (CNN), a type of feed-forward artificial neural network, used simultaneously for feature learning and classification. We use CNNs in Chapter 4 for viewpoint representation and Chapter 5 for object representation. Next, in section 2.5.2 we explain convolutional auto-encoders, another type of artificial neural networks, used for training efficient coding. Such model is used in chapter 6 for encoding face attributes.

Finally in section 2.6 support vector machines are introduced as a supervised learning model. SVM efficiently performs classification and is used in chapters 3, 4, and 5.

2.1 Bag of Visual Words

Bag of visual words (BoW) is a simplified image representation first introduced by [Sivic and Zisserman, 2003] in 2003. The idea comes from bag of words representation in text retrieval context. BoW has been one of the most commonly used representation in object recognition for almost a decade, applied in object recognition, image retrieval, action recognition and so on.

The main idea of BoW is to summarize the statistics of detected features in an image. To this end, first, a set of N local descriptors (like SIFT [Lowe, 2004]) $\mathbf{X} = [x_1, x_2, ..., x_N] \in \mathbb{R}^{D \times N}$ are extracted from every image (D is dimensionality of each descriptor). Then they are clustered to several local patches (abstraction process). This is done by building a visual dictionary of size M, $\mathbf{B} = [b_1, b_2, ..., b_M] \in \mathbb{R}^{D \times M}$, using clustering of local descriptors from training images. The center of each cluster is called a visual word which represents the characteristics of an image patch. Then each descriptor is assigned to the nearest visual word in the dictionary (*i.e.* each descriptor is converted into a code vector in which only one of its elements is non-zero). This coding is called Vector Quantization (VQ) since it quantizes each descriptor to a visual code. It can be expressed in following least-square fitting problem:

$$\min_{\mathbf{C}} \sum_{i=1}^{N} ||x_i - \mathbf{B}c_i||^2 \quad s.t.\ ||c_i||_0 = 1,\ ||c_i||_1 = 1,\ c \succeq 0,\ \forall i \tag{2.1}$$

Where $\mathbf{C} = [c_1, c_2, ..., c_N]$ is a set of code vectors for $\mathbf{X}$. The first constraint guarantees only one non-zero element in each code vector c_i and the second and third constraint means that the coding weight should be 1.

Finally, the BoW is simply a histogram describing the frequency of observing visual words in the image regardless of their position or equivalently a sum-pooling over code vectors $\mathbf{C}$. A simple illustration of this method is shown in Figure 2.1.

BoW representation is remarkably good for image-level object recognition. However, the drawback of such representation is that it is very hard to localize the object after identifying its presence since it does not encode any spatial information.

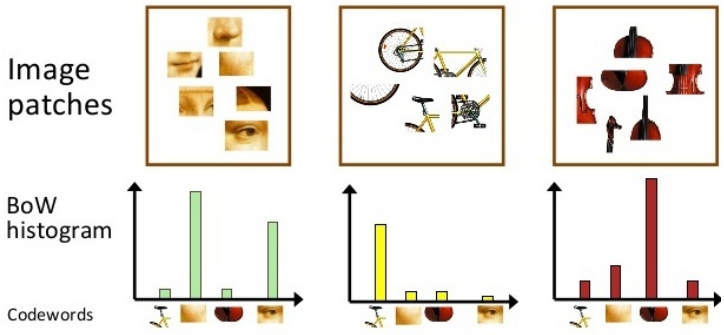

Figure 2.1: Illustration of BoW representation. Figure is borrowed from [Hoiem and Huang, 2015]

2.2 Locally-constrained Linear Coding

Locality-constrained Linear Coding (LLC) [Wang et al., 2010] is a simple and effective coding scheme that can replace VQ coding in BoW. It focuses more on the locality property than sparsity while coding each descriptor. Same as previous, after extracting descriptors and building the dictionary, LLC uses the locality constraints to project each local descriptor into its local-coordinate system.

$$\min_{\mathbf{C}} \sum_{i=1}^{N} ||x_i - \mathbf{B}c_i||^2 + \lambda||d_i \odot c_i||^2 \quad s.t.\ \mathbf{1}^T c_i = 1,\ \forall i \tag{2.2}$$

where $\odot$ denotes the element-wise multiplication, and $d_i \in \mathbb{R}^M$ is the locality adaptor as defined

$$d_i = exp(\frac{dist(x_i, \mathbf{B})}{\sigma}) \tag{2.3}$$

where $dist(x_i, \mathbf{B})$ is the Euclidean distance vector between x_i and all elements of B and σ is for adjusting the weight decay speed.

Using LLC coding, each descriptor is more accurately represented by multiple bases (proportional to their similarity to the input descriptor) and the correlations between similar descriptors are captured by sharing bases. This is the difference between VQ coding and LLC coding, as shown in Figure 2.2. Afterwards, the present codes in an image are integrated by max-pooling to generate the final image representation.

While VQ coding in combination with sum-pooling requires nonlinear classifiers to achieve good performance, LLC computes locally-constrained codes which in combination with max-pooling results in remarkably good performance even when using a linear classifier. Using a linear classifier makes LLC scalable since a nonlinear classifier has to afford additional computational complexity both in training and test. A significant benefit of linear classifiers is that they are very efficient to evaluate and efficient to learn (linear in the number of training samples).

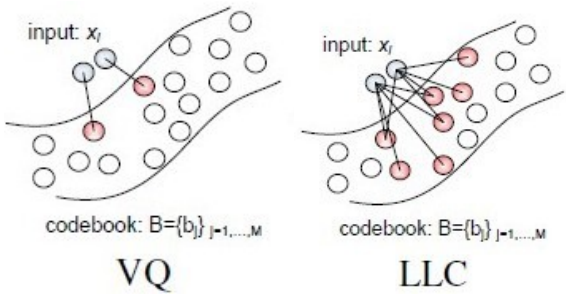

Figure 2.2: Comparison between VQ and LLC. The selected bases for representation are highlighted in red. The figure is borrowed from [Wang et al., 2010].

2.3 Fisher Encoding

The aim of Fisher Encoding [Perronnin et al., 2010] is to summarize the statistics of detected local features in an image using a vectorial representation. Similarly to BoW, it assigns local descriptors to elements in a visual dictionary. However, there are some differences. First, rather than clustering, it uses a probabilistic dictionary obtained using Gaussian Mixture Models (GMM). Second, rather than storing visual word occurrences only (zero-order statistics), this representation stores also higher order statistics. This is done by a different coding function that is used to code local descriptors. More specifically, FV first trains a Gaussian mixture model as a generative model then codes a local descriptor by capturing its deviation from the GMM parameters $\lambda_j = \{w_j, \mu_j, \Sigma_j, j = 1..M\}$ where w_j, μ_j, Σ_j are weight, mean, and diagonal covariance of the j-th mixture model respectively and M is the number of mixtures. The deviation is measured by computing the gradient of each local descriptor with respect to mean and

covariance of generative model.

$$\psi(x_i;\lambda_j) = [\frac{1}{\sqrt{w}}(\frac{x_i-\mu_j}{\sigma_j}), \quad \frac{1}{\sqrt{2w}}(\frac{(x_i-\mu_j)^2}{{\sigma_j}^2}-1)] \tag{2.4}$$

Then low-level features statistics with respect to component j are aggregated using weighted-average pooling:

$$G^{\mathbf{X}}_{\lambda_j} = \frac{1}{N}\sum_{i=1}^{N}\alpha_j^i\psi(x_i;\lambda_j) \tag{2.5}$$

where α_j^i is the soft weight of the i-th descriptor to the j-th Gaussian component.

Finally the computed gradients with respect to all model parameters are concatenated which leads to a vectorial representation called Fisher Vector (FV).

$$FV(\mathbf{X}) = [G^X_{\lambda_1}, ..., G^X_{\lambda_M}]. \tag{2.6}$$

Similarly to BoW and LLC, this representation is spatially order-less. Different than BoW and similarly to LLC, FV performs well even with simple linear classifiers. However while BoW image representation is typically quite sparse, the FV is almost dense and its dimensionality is much higher that BoW representation.

2.4 Spatial Pyramid

As mentioned previously, the BoW, LLC and FV do not encode any spatial information in their representations. Spatial pyramid (SP) [Lazebnik et al., 2006] tackles this limitation by partitioning the image into increasingly finer grids and then representing each grid independently. The resulting spatial-coded representation is an extension of an order-less representation and showed better performance in object recognition. A simple illustration of SP is shown in Figure 2.3.

2.5 Convolutional Neural Networks

Convolutional neural networks (CNN) [LeCun et al., 1998] are a special class of artificial neural networks which are inspired from the visual cortex of animals and are enjoying great success nowadays in the field of object detection and

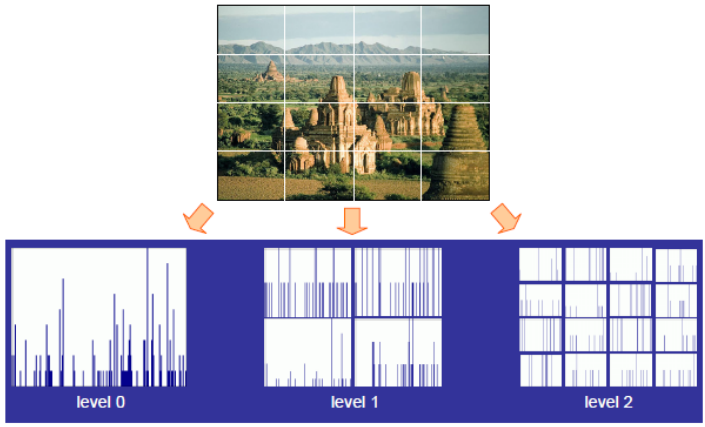

Figure 2.3: Illustration of a 3-level spatial pyramid encoding. The image is divided in three different levels of resolution and for each, a representation is computed. The final representation is concatenation of all of the sub-region representations.

classification. CNNs consist of multiple layers stacked in such a way that the output of one layer serves as the input to the next layer. CNNs allow the learning of the deep networks by the aid of introducing three main ideas which are: local receptive fields, parameter sharing and pooling.

Local receptive fields: When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. To exploit spatially local correlation, in the CNNs the connections are made in the small, localized regions of image. A small region of input neurons is connected to each neuron of the next hidden layer. That small window is referred as the local receptive field for hidden neurons. This local window is moved across the entire image in a sliding window manner with a fixed stride. The filters in the convolutional layers are applied locally in the small regions of the images, giving the local representations of the region. While going deeper in the network, representations become more global as the receptive field increases in each next layer. Therefore at the lower layers convolutional neural networks are able to create simple representations of small parts of the inputs whereas higher layers generate descriptors of larger regions of the input.

Parameter sharing: In the CNNs, the same filter (having same weights and bias) is applied across the entire input field resulting in the feature maps of the

same size as the original input. The filters share the weights but differ only in the sense that they are applied at different locations in the image. This makes sense because features learnt at one place in image can be useful at other places in the image. The main advantage of sharing the same filters is that it makes the features translational invariant, so changing the position of an object in the image does not affect the performance of the detector. As the network learns only one type of features, still the detector is not very robust, therefore additional filters are added which learn different types of features, making the network more robust. The other advantage of shared filters is that it significantly reduces the number of parameters that need to be learnt at each layer as compared to a conventional neural network.

Pooling: The last main thing which differentiates the CNNs from fully connected neural networks is the introduction of pooling layers. The main aim of this layer is to simplify the output of convolutional layers by non-linear down-sampling of the information involved in each local region. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. It progressively reduces the spatial size to reduce the amount of parameters and hence to also control overfitting.

Finally, after several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Figure 2.4 shows a graphical depiction of a CNN model. Such architecture can be consider as a feature learning algorithm: convolutional layers are exceptionally good at finding good features in images to the next layer to form a hierarchy of nonlinear features that grow in complexity (e.g. [blobs, edges] -> [noses, eyes, cheeks] -> [faces]). The final layer(s) use all these generated features for classification or regression.

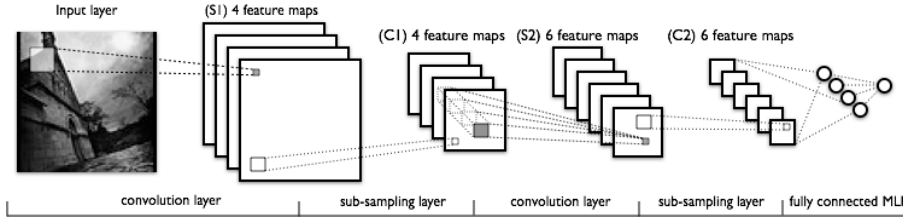

Figure 2.4: Illustration of a typical CNN model called LeNet [LeCun et al., 1998]. The lower-layers are composed of alternating convolution and pooling layers while the upper-layers are fully-connected.

2.5.1 Modern CNN Architectures

LeNet [LeCun et al., 1998] was one of the first CNN architecture used in the computer vision applications, named after one of its author Yann LeCun. It was designed for the hand and machine written characters recognition. The architecture of LeNet is shown in the Figure 2.4. Most of the recent CNNs architectures are inspired by this architecture. LeNet consists of five hidden layers after the input layer. First hidden layer is a convolutional layer following a pooling layer. This structure is repeated one more time and the last hidden layer is a fully connected layer.

To extract the features from this network, the image is first converted to a predefined fixed size and then is convolved with the learnt filters. The output of this step which is referred as the feature map or activation map or activation map is passed through the pooling layer to reduce the parameter space (a mechanism to control over-fitting). For pooling, mostly max-pooling is used *i.e.* it uses the max operation inside a fixed-size filter to sub-sample the feature map. Another important element in a CNN architecture is the activation function. To make the network able to learn nonlinear models, non-linearity is introduced into the network through the activation functions. Sigmoid, Hyperbolic tangent and Rectified Linear Unit (ReLU) are the commonly used activation functions. The ReLU function is defined as $f(\mathbf{x}) = max(0, \mathbf{x})$. The main advantage of using the ReLU function is that it is empirically proved suitable for avoiding over-fitting. Also, as the ReLU unit requires only a comparison, it allows the fast training of the network on the complex datasets. Moreover, based on its output formula, it gives a sparse feature map.

During recent years many CNN models are proposed for different tasks in computer vision such as classification and detection. Most of them are based on the basic LeNet. Here, we briefly describe three modern, popular CNN models:

AlexNet AlexNet [Krizhevsky et al., 2012] is considered as a major breakthrough in computer vision. AlexNet won the Imagenet large scale visual recognition challenge (ILSVRC) 2012 by achieving the top-5 error rate of 15.3% for classification task by a significant margin (the second method achieved top-5 error rate of 26.2%). It is based on the LeNet, but uses more convolutional layers which are stacked on top of each other (see Fig. 2.5). It consists of seven hidden layers of which the first five are convolutional layers and the last two are fully connected layers. It consists of about 650K neurons, 630M connections and 60M parameters.

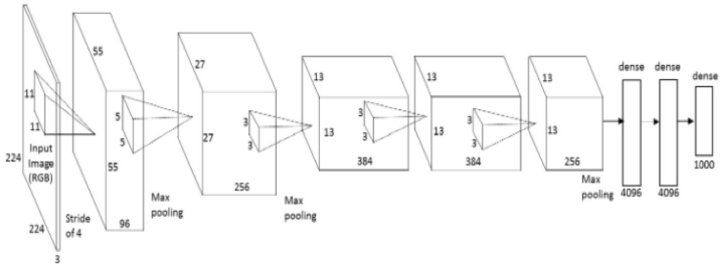

Figure 2.5: Alexnet architecture. The image is borrowed from https://goo.gl/jkJLcK

DeCAF DeCAF [Donahue et al., 2013] is an efficient implementation of AlexNet that closely follows its architecture and training protocol with the exception of two small differences in the input data. First, it ignores the image original aspect ratio and warps it to 256×256 rather than resizing and cropping to preserve the proportions. Secondly, it does not perform the data augmentation trick of adding random multiples of the principle components of the RGB pixel values throughout the dataset, proposed as a way of capturing invariance to changes in illumination and color

VGGNet VGGNet [Simonyan and Zisserman, 2015] is a deep implementation of CNN having a simple and repeating structure. VGGNet-16 contains 16 convolution layers and two fully connected layers. The simplicity is provided by repeating the structure of 2 convolution layers followed by a max pooling layer (see Fig. 2.6). The use of deeper networks increases the performance of network in the classification and detection tasks but it also increases the memory requirements. The number of parameters involved in this network is more than 140M.

2.5.2 Convolutional Auto-encoders

A convolutional autoencoder (CAE) [Masci et al., 2011] is a feedforward neural net which is very similar to the CNNs, with an input layer, an output layer and one or more convolutional layers connecting them. The difference between CAEs and CNNs, is that in an autoencoder, the output layer has the same number of nodes as the input layer, and that, instead of being trained to predict the target value, it is trained to reconstruct their own inputs. More specifically, a CAE is composed of two parts: convolution and deconvolution networks. The convolution network corresponds to a feature extractor (encoder)

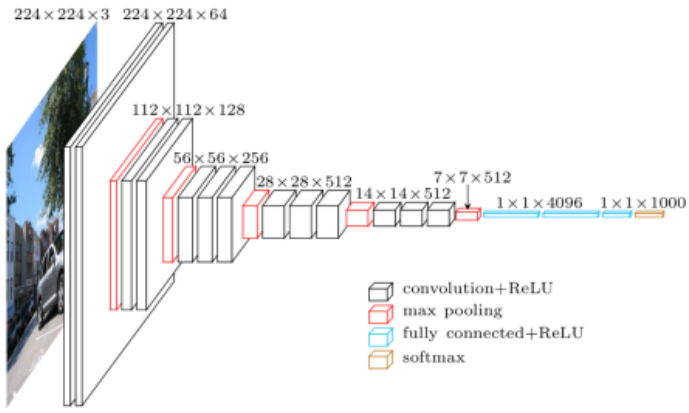

Figure 2.6: VGG architecture. The image is borrowed from https://goo.gl/4WThNZ

that transforms the input image to a multidimensional feature representation (typically the dimension of the code is much less than the input), whereas the deconvolution network produces the input from the encoder (decoder). A deconvolution layer is responsible to densify a set of feature maps through convolution-like operations since the output of an unpooling layer is sparse feature maps. So in deconvolutional layers padding is removed from the output rather than added to the input (as is done in convolutional layers). In this sense, they are doing convolution inversely. A sample CAE is shown in Figure 2.7.

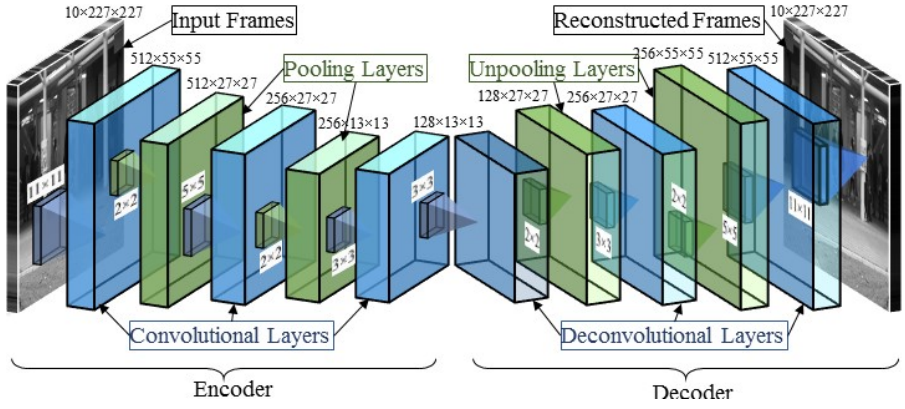

Figure 2.7: Structure of a fully convolutional autoencoder. The image is borrowed from [Hasan et al., 2016].

2.6 Support Vector Machines

Support Vector Machines (SVMs) are the most commonly used supervised learning method in the field of computer vision. SVM is a non-probabilistic parametric classifier which given a set of training examples, each marked for belonging to one of two categories, its training algorithm builds a model that assigns new examples into one category or the other. While classifying two classes, there are many hyperplanes that might classify the data. In the SVM objective function, the best hyperplane is defined as the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized.

SVM at the beginning was only proposed for two-class tasks. For a multiclass classification, [Crammer and Singer, 2001] proposed a multiclass SVM in a single optimization problem. However, for a multi-class classification, the dominant approach is to decompose the single multiclass problem into multiple binary classification problems in a *one-versus-all* manner (*i.e.* distinguishing between one of the labels and the rest of labels).

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the *kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces. In this case, the original finite-dimensional space is implicitly mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function.

Chapter 3

Action Representation for Video Classification and Segmentation

In this chapter we investigate representations in the context of action recognition and video segmentation. We show that splitting the representation of a video into action-related foreground and action-unrelated background can improve recognition performance. This chapter directly addresses research question 1 by proposing different methods to integrate, at different levels, action recognition and binary motion segmentation. At the end of this chapter, we will understand how the quality of motion segmentation affects the task of action recognition. Work covered in this chapter is based on:

- Ghodrati, A., Pedersoli, M. and Tuytelaars, T, *Coupling video segmentation and action recognition*, In IEEE Winter Conference on Applications of Computer Vision (WACV), 2014.

3.1 Introduction

Alongside category-level object recognition and detection, action recognition is, arguably, one of the big computer vision challenges. The first successes in this domain were obtained by realizing that actions can be considered as spatio-temporal objects and, therefore, the wide gamut of methods developed

for object recognition and detection can be extended to action recognition, often in a relatively straightforward way. As one particularly successful example, building on spatio-temporal versions of 2D interest point detectors [Laptev and Lindeberg, 2003], bag-of-visual words based image classification has been applied to action recognition [Laptev et al., 2008, Wang et al., 2009].

However, the analogy between actions and spatio-temporal objects only holds up to some point. There are also important differences, that lead to particular challenges and limitations as to what can be achieved simply by following the analogy:

1. First, actions typically exhibit *larger variability* (at least, as soon as the switch to realistic datasets was made). Objects vary in appearance due to varying viewpoints, changing backgrounds, partial occlusions and within-class variability. Actions additionally vary because they are performed by different people, at different speeds and in different ways. Moreover, also the camera motion is typically uncontrolled.

2. At the same time, the *number of training examples* is usually much smaller. This is due to the rather cumbersome process of data collection and due to the memory usage of datasets quickly growing when dealing with video. This is a difficult combination, as the larger variability in the data requires more complex models, which in turn require more data to avoid overfitting. To overcome this problem, we either need strong priors or robust methods that are designed to already compensate for a great deal of the variation.

3. Additionally, actions are often *not well delineated by bounding boxes* – especially if the action location changes over time, such as for *walking* or *running*, or when the camera is not static. As a result, the popular sliding window based scheme cannot be used as effectively for actions as for objects. Moreover, it also becomes more expensive, since the search space increases from 3D (x, y and scale) to 5D (x, y, t, spatial and temporal scale).

4. Finally, actions are *not localized as precisely* as objects. Different annotators often do not agree on the actual extent of an action. This holds both for the temporal delineation (When does the action start? When does it end?), as well as for the spatial delineation (Does the action include the whole actor or only part? Should objects that are involved be included as well? etc.).

At the same time, the spatio-temporal nature of actions may also hold opportunities. In particular, video segmentation is often more reliable than image segmentation (i.e., more consistent with object boundaries). This is due

to the fact that motion brings an important additional cue to delineate the objects (or actors) from the background. In this chapter, *we investigate whether video segmentation can be exploited for improved action recognition.*

In this context, it is striking that many works on video segmentation [Brox and Malik, 2010, Lezama et al., 2011] and action recognition [Gaidon et al., 2012, Raptis et al., 2012] build on the same set of low level features, known as *trajectories* [Wang et al., 2011]. Trajectories are sampled patches that are tracked over several frames, following the underlying motion of the object or scene. These features are extracted densely over multiple spatial scales and described based on their shape, appearance and motion information. Video segmentation can be done by clustering extracted trajectories to foreground and background. Moreover, higher-level representation that we define it in section 3.3.2 can be obtained from these trajectories. Such higher level representation can also be used for video segmentation (see Fig. 3.1). Using trajectories, state-of-the-art results have been reported both for action recognition [Gaidon et al., 2012, Wang et al., 2011] and video segmentation [Brox and Malik, 2010, Lezama et al., 2011].

We therefore start from the representation proposed in [Wang et al., 2011]. We build on trajectories as low level image representation, combine them in a simple bag-of-words (`BoW`) representation and then learn a linear support vector machine (SVM) classifier. Considering this as our baseline, we propose and evaluate several methods that integrate, at different levels, segmentation and recognition. Note that, since action localization is often ambiguous, we deliberately focus on the task of action classification rather than detection, even though the obtained segmentations obviously also provide us with some localization information.

We focus on the following schemes:

1. **Segmentation.** We split the representation of a video into action-related foreground and action-unrelated background, and build separate `BoW` for each of them (see section 3.3.2). We experiment with both bottom-up as well as top-down segmentation, where the latter is explicitly geared towards actions.

2. **Co-segmentation.** Co-segmentation is defined as the task of jointly segmenting "something similar" in a given set of images/videos. In section 3.3.3 we encourage consistent segmentations over multiple videos via co-segmentation. This results in better segmentation and therefore improved recognition accuracy.

3. **Iterative learning.** A better segmentation is likely to produce better recognition. At the same time, a better recognition can help to induce

better segmentation. In section 3.3.4 we explain how to iteratively refine both tasks in a coupled learning framework.

4. **Kernels.** As amply demonstrated in the literature, the bag of words (`BoW`) representation can be improved by mapping the original feature space with a non linear kernel. In section 3.3.5 we apply this idea to our problem and describe its advantages and disadvantages.

The remainder of this chapter is organized as follows. Section 3.2 discusses related work. Next, we describe the various schemes for action classification (Sec. 3.3) and our experimental results (Sec. 3.4). Section 3.5 concludes the chapter.

3.2 Related Work

Combined segmentation and recognition. Within the field of object recognition and image understanding, it is widely accepted that, for good results, segmentation and object recognition should go hand in hand and be solved simultaneously. A purely low-level image segmentation is bound to fail at the semantic level. Still, not many methods bring this into practice. Most works on object recognition simply use bounding boxes instead of segmenting out the object of interest, see e.g. [Bilen et al., 2011]. An exception to this trend is the work on class-independent object detection, where segmentation plays an important role as one of the few consistent cues over object categories [Alexe et al., 2010, Endres and Hoiem, 2010]. Also Rosenfeld and Weinshall [Rosenfeld and Weinshall, 2011] explicitly link segmentation to object recognition, by learning to extract the foreground mask from training images. Object recognition methods using superpixels are probably the closest equivalent to our action recognition approach. For instance, Fulkerson *et al.* [Fulkerson et al., 2009] propose a method for class segmentation and object localization with superpixel neighborhoods. However, their goal is mostly a high quality image segmentation, rather than a better performance in object detection. In the context of action recognition, Ullah *et al.* [Ullah et al., 2010] experiment with segmentations based on motion, action, humans or objects. Action-based segmentation yields the best recognition results. However, this scheme relies on an external dataset of static images of the same set of actions, which may not always be available, and makes a fair comparison with alternative schemes difficult. Hoai *et al.* [Hoai et al., 2011] also looked into the problem of joint segmentation and classification of human actions. However, their work only considers temporal segmentation.

Co-segmentation. Another line of work related to ours is co-segmentation. Hochbaum and Sing [Hochbaum and Singh, 2009] noticed that for co-

segmentation, instead of penalizing for variations between foreground descriptors, a similar effect can be achieved by rewarding consistency between them. This leads to a tractable optimization problem that can be solved efficiently. Vezhnevets *et al.* [Vezhnevets et al., 2011] propose a weakly supervised approach that segments images jointly by learning a CRF model connecting all superpixels from all training data in a data-driven fashion. Besides the task, our method differs from theirs because they assume coherency in both background and foreground superpixels, while our model only captures foreground coherency between videos of the same class. Prest *et al.* [Prest et al., 2012] extract a fixed number of segments in each video. They assume one of them is the object and the others are background. They minimize an energy defined jointly over all training videos to select one segment per video. Finally, Rubio *et al.* [Rubio et al., 2012] extended standard co-segmentation to videos.

Iterative methods. Deselaers *et al.* [Deselaers et al., 2010] propose a model that iteratively localizes the objects and learns class-specific appearance and shape. For each image they find the top 100 bounding boxes that are likely to be an object and select the one that best optimizes an energy function defined globally over all training images. Shapovalova *et al.* [Shapovalova et al., 2012] have proposed a similar method to localize consistent foreground 3D boxes across an action class. Our proposed method uses a similar iterative scheme to simultaneously segment the foreground and learn its appearance. However, in our case the selected foreground is not a 3D box, but a more expressive combination of trajectory-groups. Lan *et al.* [Lan et al., 2011] propose a spatial model that considers action parts as latent variables. They jointly localize and recognize actions using a figure-centric representation. Raptis *et al.* [Raptis et al., 2012] cluster trajectories into groups, with each group a candidate for the parts of an action. This is a part-based model incorporating both individual appearance and motion constraints as well as spatio-temporal pairwise constraints. It learns to localize not only the action, but also its constituent parts. We also start from groups of trajectories, so our initial representation is very similar. However, in contrast to [Raptis et al., 2012], we do not attempt to exploit the configuration of these groups. That is because that the amount of training data in action recognition datasets that were available during this work was insufficient to localize and learn action subparts. Therefore, we construct a single bag-of-words descriptor for all the foreground groups and one for all the background groups. This is more robust than treating each group separately, as in [Raptis et al., 2012].

Trajectories-based action recognition. In the context of action recognition, trajectories have recently been used in various ways, also going beyond the simple bag-of-words scheme. The most relevant for us are probably [Raptis et al., 2012] (see above), [Gaidon et al., 2012] and [Sapienza et al., 2012]. Gaidon

et al. [Gaidon et al., 2012] perform a hierarchical clustering of the trajectories and propose a kernel that computes the structural and visual similarity of two hierarchical decompositions. They obtained great results. Our method differs from [Gaidon et al., 2012] in that our segmentation into foreground and background is not just driven by similarity between groups, but also by top-down cues. Finally, Sapienza *et al.* [Sapienza et al., 2012] also start from trajectories and learn discriminative action subvolumes in a multiple instance learning framework.

3.3 Coupled Segmentation and Recognition

In this section we define several schemes that can be used to improve action recognition over our baseline model. First we describe the basic framework (section 3.3.1). In section 3.3.2 we propose methods based on segmentation whereas in section 3.3.3 we add co-segmentation to enforce similarity among foreground models of the same object class. Then in section 3.3.4 we present a model which simultaneously localizes actions and learns class-specific foreground and background models. Finally in section 3.3.5 we consider a non-linear representation of our model.

3.3.1 Baseline

Throughout our work we use the *dense trajectories* proposed by [Wang et al., 2011] as the features. Using these features, they obtain state-of-the-art results with a simple bag-of-words representation. In a dense grid, they tracked each sampled point $P^t = (X^t, Y^t)$ at frame t over the next L frames by median filtering in the optical flow field. Points of subsequent frames are concatenated to form a *trajectory* $T_i = (P^t, ..., P^{t+L})$. When the trajectory reaches length L, it is added to the trajectory pool. Then, a new point is picked at the same location for tracking, if no other tracking point is found in the neighborhood. With a median tracker, in case of long-term tracking of fast motions, it is likely that the trajectory will drift from its initial location. This occurs in particular when a foreground pixel is crossing the background causing the tracker to leave the initial point on the background and to start tracking the foreground instead. However, for fixed length trajectories, as in our case, we did not frequently observe this phenomenon and the resulting trajectories are good enough for clustering. For each trajectory, a local descriptor is computed around its 3D volume. As in [Wang et al., 2011], we use histograms of oriented gradients (HOG) to encode static appearance information, histograms of optical flow (HOF) to encode absolute motion information of trajectories, and motion

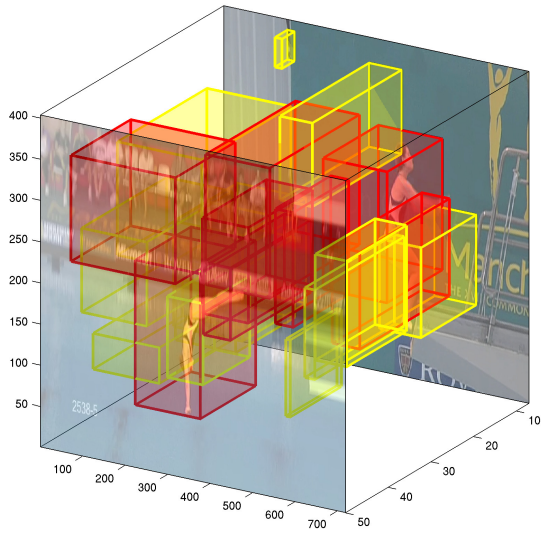

Figure 3.1: Video segmentation: we cluster the video into several trajectory-groups (colored boxes). Then each one is assigned to foreground (red boxes) or background (yellow boxes) based on top-down cues.

boundary histograms (MBH) to capture relative motion (i.e., discard constant motion information). We build three codebooks, one for each descriptor type, using K-means clustering ($W = 4000$) on a subset of $100,000$ randomly sampled features. The final representation is obtained using a locally linear constrained coding (see section 2.2) and max-pooling [Wang et al., 2010]. For the `BoW` baseline, we concatenate for each video the three histograms and learn a one versus all linear SVM. The trade-off between loss and regularization, C, is set to 100 for this and for all the following SVM training in this chapter.

3.3.2 Segmentation

There are several algorithms for segmenting a video. Most of them generate an over-segmented video volume, which is further used for grouping into super-regions [Xu and J.Corso, 2012, Grundmann et al., 2010, Brox and Malik, 2010]. We use this video segmentation in two settings: first, as a fully bottom-up foreground/background segmentation, and second, as an initial over-segmentation of the video in what we call *trajectory-groups*, that will serve later as intermediate-level representation for our top-down segmentation.

A trajectory-group is a set of spatially close trajectories with similar motion patterns. To group trajectories, we employ three criteria to define a distance measure between them, ensuring that we are clustering trajectories that are spatially close to each other, that move together consistently and that co-exist in a time interval. For trajectories T_i and T_j, the distance is defined as

$$d(T_i, T_j) = d_S(T_i, T_j) \times d_H(T_i, T_j) \times d_T(T_i, T_j)$$

where d_S is the maximum Euclidean distance between points of the two trajectories in the same frame (if they have overlap), d_H is the Euclidean distance between HOF descriptors and d_T measures the time intersection of trajectories. If there exists a time interval over which both trajectories co-exist, then $d_T(.) = 1$; otherwise, it is equal to the temporal distance (number of frames) between the trajectories. Note that this trajectory distance measure is similar to the ones used in [Raptis et al., 2012, Lezama et al., 2011, Brox and Malik, 2010]. We turn the distances into affinities using standard exponential $w_{i,j} = exp(-\gamma d(T_i, T_j))$ with fixed scale $\gamma = 0.1$.

For each video, we randomly select 6000 trajectories and build a fully connected graph with nodes corresponding to trajectories and $w_{i,j}$ the weight of the edge between nodes i and j, forming an $n \times n$ edge affinity matrix with n the number of nodes. To assign to each node a label $l_i \in L$ from label set $L = \{l_1, ..., l_K\}$ in an unsupervised manner, we employ the method proposed in [Shi and Malik, 2000] which minimizes the normalized-cut of the graph. Each of these K clusters is a trajectory-group, described by h_i, as shown in Fig. 3.1.

Bottom-up By choosing $K = 2$, our algorithm behaves as in [Brox and Malik, 2010] producing a segmentation of the video into foreground and background motion. To distinguish between foreground and background cluster we need to compute their overlap with the ground truth annotation, assigning the cluster with higher overlap to foreground[1]. The final descriptor $\mathbf{H}$ of each video is the concatenation of LLC coded of the three previously described features for foreground (h_f) and background (h_b). We refer to this setting as `Bottom-Up` segmentation .

Top-down In this setting we first over-segment the video into K trajectory-groups (with $K > 2$). Then, we assign each trajectory-group to either foreground or background, based on a learnt model. For training we use the bounding box annotations indicating the location of the actions to label trajectory-groups to

[1] we use the ground truth annotations of test data only for this baseline, that corresponds to an upper bound of what can be achieved with bottom-up segmentation. We do not use ground truth annotations of test data for our proposed method

foreground or background. The trajectory-group i is assigned to foreground, i.e. $x_i = +1$, if at least one quarter of trajectories inside it, is assigned to the foreground; otherwise we set $x_i = 0$. A trajectory is assigned to the foreground if the majority of its points lie in the foreground bounding box. Given a set of training trajectory-groups from each action, $\mathbf{h} = \{h_1, ..., h_r\}$ and their corresponding labels $\mathbf{x} = \{x_1, ..., x_r\}$, we learn the linear model Θ for action-related (foreground) against action-unrelated (background) using a linear SVM.

As we use a linear model, $\Theta^T h_i$ is the learned scoring function for the trajectory-group h_i and $p(h, \Theta)$ is a function that maps that score to $[0, 1]$. For each video we want to find the best configuration of binary labels $\mathbf{x}$ in terms of the segmentation score. Hence, we define the cost function $\sum_{i=1}^{K} \Omega(x_i, h_i, \Theta)$ where

$$\Omega(x, h, \Theta) = \begin{cases} p(h, \Theta) & \text{if} \quad x = 0 \\ 1 - p(h, \Theta) & \text{if} \quad x = 1 \end{cases} \tag{3.1}$$

It is easy to show that

$$\min \sum_{i=1}^{K} \Omega(x_i, h_i, \Theta) =$$

$$\min \sum_{i=1}^{K} p(h_i, \Theta)(1 - x_i) + (1 - p(h_i, \Theta))x_i =$$

$$\sum_{i=1}^{K} \min \left[(1 - 2p(h_i, \Theta))x_i \right]. \tag{3.2}$$

So the label of each trajectory-group can be obtained with an independent minimization: if $p(h, \Theta) < \frac{1}{2}$ then h is assigned to background otherwise it is considered foreground.

So far, each video is a set of trajectory-groups and their corresponding predicted binary labels. The final representation of a video is the concatenation of the foreground and background histograms, where each of these is obtained by summing the histograms of foreground and background trajectory-groups respectively: $\mathbf{H} = [\mathbf{H}_f, \mathbf{H}_b]$ with:

$$\mathbf{H}_f = \frac{1}{C_f} \sum_{i=1}^{K} x_i h_i, \quad \mathbf{H}_b = \frac{1}{C_b} \sum_{i=1}^{K} (1 - x_i) h_i \tag{3.3}$$

where C_f and C_b are the mean value of the Euclidean distance between all training samples of foreground and background respectively.

As all foreground trajectory-groups most likely represent human actions they should have quite similar appearance. Therefore, instead of learning a different segmentation model independently for each action class, we can also learn a shared model for all actions. Such segmentation is learned by extracting generic knowledge about actions and can be considered an *actionness* detector, similar to objectness [Alexe et al., 2010] or object proposals [Endres and Hoiem, 2010] in static images or in video [Stalder et al., 2012]. In the experiments we refer to `Class-specific` segmentation when using a different model for each action and `Actionness` when using a shared model for all actions. Finally, as a control, we also experiment with a `Random` segmentation, where each trajectory-group is assigned to foreground or background using a uniform probability distribution.

3.3.3 Co-segmentation

We can refine the segmentation by encouraging as foreground trajectory-groups among different videos with similar appearance. To this end, we minimize the energy function E defined globally over all training videos of class c:

$$E(\{\mathbf{x}^j\}_{j\in c};\Theta) = \sum_{h_i^j \in G_v} \Omega(x_i^j, h_i^j, \Theta) - \lambda \sum_{(h_i^j, h_{i'}^{j'})\in G_e} S(h_i^j, h_{i'}^{j'}) x_i^j x_{i'}^{j'}. \quad (3.4)$$

The graph G is built in a data-driven fashion by selecting all trajectory-groups from all the videos of class c as nodes G_v, and by connecting the k most similar ones to create the edge set G_e.

The unary term Ω is defined as in Eq.(3.1) and considers the cost of the segmentation algorithm (e.g. `Actionness`).

The pairwise cost is defined by S which is a similarity function between two trajectory-groups and is computed as: $S(h_1, h_2) = \langle h_1, h_2 \rangle$ and rescaled to $[0, 1]$. The pairwise term encourages coherent foregrounds to take the same label since foreground trajectory-groups with high similarity will serve as a reward to the cost function E. Note that for a fully connected graph as in [Hochbaum and Singh, 2009], the pairwise potential turns to $\sum_{j,j'} \sum_{l,m} \left\langle h_l^j, h_m^{j'} \right\rangle x_l^j x_m^{j'} = \sum_{j,j'} \left\langle H_f^j, H_f^{j'} \right\rangle$, the similarity between the foreground descriptors. Our pairwise term differs from [Vezhnevets et al., 2011] as we encourage similarity only on foreground regions, whereas in [Vezhnevets et al., 2011] the pairwise term encourages also similarity between background regions.

λ is a parameter to balance between the unary and pairwise term. The energy function E is sub-modular since the pairwise potential is always negative when $x_i^j = 1$ and $x_{i'}^{j'} = 1$. Thus E can be minimized efficiently employing graph-cut algorithm [Boykov et al., 2001]. Note that when $\lambda = 0$ the pairwise term vanishes and E depends only on the unary term as defined in Eq.3.1. In the experimental results we test the impact of adding co-segmentation to the previously considered segmentations. Note that one can also add pairwise constraints within a single video, e.g. between nearby trajectory-groups. We tried this as well, but did not observe any significant improvement in the results.

3.3.4 Iterative Learning

We believe that segmentation and recognition can help each other as a better segmentation leads to better recognition models and vice versa. The methods described so far first solve the segmentation and then use its output during action classification. Here, we propose an iterative learning scheme that alternates between segmentation and recognition. The method at the first step segments an action by finding the labeling vector $\mathbf{x}$ for every video and then as second step it finds the new appearance models for foreground and background given the current segmentation. These new models are then used in the next iteration to get better segmentation and so on.

Considering that an initial global model for foreground (Θ_f) and background (Θ_b) action classification is given, the goal of the first step is to yield the best labeling $\mathbf{x}$ for each video of a certain action. Let Λ_f and Λ_b be the scoring function associated to the foreground and background descriptors $\mathbf{H}_f$, $\mathbf{H}_b$ of a video. Our goal is to find a configuration of latent variables $\mathbf{x}$ that maximizes the classification function

$$\Lambda_f(\mathbf{x}; \mathbf{H}_f, \Theta_f) + \Lambda_b(\mathbf{x}; \mathbf{H}_b, \Theta_b) \tag{3.5}$$

i.e. discriminates as much as possible one class from the others. The fact that $\mathbf{H_f}$ ($\mathbf{H_b}$), the video-level representation of foreground (background), is coupled to trajectory-group labels $\mathbf{x}$, makes the maximization hard since Θ depends on the complete descriptor $\mathbf{H}$. However, if we choose linear models for the scoring

function then:

$$
\begin{aligned}
argmax\left(\Lambda_f(\mathbf{x};\mathbf{H}_f,\Theta_f)+\Lambda_b(\mathbf{x};\mathbf{H}_b,\Theta_b)\right) = \\
argmax\left(\Theta_f^T\mathbf{H}_f+\Theta_b^T\mathbf{H}_b\right) = \\
argmax\left(\sum_{i=1}^{K}\Theta_f^T h_i x_i+\sum_{i=1}^{K}\Theta_b^T h_i(1-x_i)\right) = \\
argmax\left(\sum_{i=1}^{K}(\Theta_f-\Theta_b)^T h_i x_i+\sum_{i=1}^{K}\Theta_b^T h_i\right) \qquad (3.6)
\end{aligned}
$$

Where $\sum_{i=1}^{K}\Theta_b^T h_i$ is independent of parameter x_i and can be discarded.

In our case, each trajectory-group is described by the LLC representation, but then, to make the model linear, the final model is the sum of the score of each trajectory-group, so that the linearity is preserved. In this way, to assign the latent variables $\mathbf{x}$ for a given video, it is enough to score each trajectory-group independently using $\Theta_f-\Theta_b$.

At the second step, given the labels $\mathbf{x}$ for each trajectory-group, we want to update the global models Θ_f,Θ_b. These linear models can be concatenated and make global model $\Theta_g=\left[\Theta_f^{HOG},\Theta_f^{HOF},\Theta_f^{MBH},\Theta_b^{HOG},\Theta_b^{HOF},\Theta_b^{MBH}\right]$. However, optimizing Θ_g is no longer convex for negative samples since it depends on the segmentation $\mathbf{x}$. Therefore a standard linear SVM would produce poor results. Instead, we use the strategy proposed in [Felzenszwalb et al., 2010]: we fix the values for the positive examples, while for the negative ones we iteratively search for new configurations of latent variables until these do not generate any more loss.

The model computed using the iterative learning can also be used in conjunction with an independent segmentation, as those defined in section 3.3.2 or the co-segmentation introduced in section 3.3.3. While all the combinations of independent segmentation, co-segmentation and iterative learning are tested in the experimental results, here we show just the case of the combination of all

three terms:

$$E(\{\mathbf{x}^j\}_{j\in c};\Theta_g^c) = \alpha \sum_{h_i^j \in G_v} \Omega(x_i^j, h_i^j, \Theta) + (1-\alpha) \sum_{h_i^j \in G_v} \varphi(x_i^j, h_i^j, \Theta_g) - \lambda \sum_{(h_i^j, h_{i'}^{j'}) \in G_e} S(h_i^j, h_{i'}^{j'}) x_i^j x_{i'}^{j'}. \quad (3.7)$$

Here φ is the normalized score of the iterative learning score (Eq. 3.5), while Ω and S are defined as before. α is set to 0.3 in all experiments and it defines the balance between independent segmentation and the normalized scores obtained by iterative learning.

As this iterative procedure is non-convex, the final result depends on the initialization of the segmentation. In the experimental results we evaluate our procedure using three different initializations: `Random` initialization, an initialization based on `Actionness` and one based on the ground truth `Annotations` [2].

3.3.5 Non-linear Kernel

While the iterative learning is a powerful tool, it is limited to linear models. An alternative way to improve results is by mapping the features into a kernel, such that non linear classifiers can be used while keeping the learning optimization convex. Excluding the iterative learning, all the other combinations of different segmentation and co-segmentation can be used together with a kernel: we first segment video then represent foreground and background of each video separately using bag of words. Considering $\mathbf{H} = [\mathbf{H}_f, \mathbf{H}_b]$ the global descriptor of a video, it can be decomposed into 6 channels corresponding to HOG, HOF and MBH for foreground and for background. As in [Ullah et al., 2010, Wang et al., 2011], for classification, different descriptors are combined in a multi-channel approach:

$$K(\mathbf{H}_i, \mathbf{H}_j) = \sum_c exp(-\frac{1}{A_c} d_{\chi^2}(\mathbf{H}_i^c, \mathbf{H}_j^c))$$

where d_{χ^2} is the χ^2 distance measure between two histograms and A_c, the normalization factor, is the mean value of χ^2 distances between all training

[2] Note that here we only use ground truth annotations of training data, not for test data.

samples of the c-th channel. As a result, the final kernel $K(.)$ is the linear combination of each channel kernel.

We train a model for each category and classify test samples using the one-versus-all strategy. While for linear kernels the best coding is LLC, for non linear kernels we also test the method with hard quantization and average pooling. The different configurations of the method are evaluated in section 3.4.3.

3.4 Experiments

In this section, we first introduce the two standard benchmark datasets used in our experiments. Then we evaluate the different components of our approach on these datasets. Finally we compare our method with state-of-the-art methods.

3.4.1 Datasets

The **YouTube** [Liu et al., 2009] dataset contains 11 action categories: basketball shooting, biking/cycling, diving, golf swinging, horse back riding, soccer juggling, swinging, tennis, swinging, trampoline jumping, volleyball spiking, and walking with a dog. The dataset contains 1600 low quality (240×320) videos. We follow the original setup using leave-one-group-out cross validation for a pre-defined set of 25 groups. This dataset is challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background and illumination conditions.

The **UCF-Sports** [Rodriguez et al., 2008] dataset contains 10 action categories: swinging (on the pommel horse and on the floor), diving, kicking (a ball), weight-lifting, horse-riding, running, skate-boarding, swinging (at the high bar), golf swinging and walking. The dataset consists of 150 video samples, extracted from sport broadcasts. The dataset is challenging due to the large intra-class variability and cluttered background. For this dataset we follow the leave-one-video-out (LOO) strategy as in [Wang et al., 2011] as well as the single split proposed in [Lan et al., 2011]. Note that for the LOO evaluation we include in the training data also the horizontally flipped version of each video as in [Wang et al., 2011], while for the single split we do not as in [Lan et al., 2011].

For both datasets, we report average accuracy over all classes, as also done in [Gaidon et al., 2011, Wang et al., 2011]. We use the YouTube dataset that contains many more videos and has more stable results to evaluate individually

Segmentation	Acc.	std.
`GT trajectory`	90.2%	3.2
`GT trajectory-group`	88.5%	3.2
`BoW`	83.5%	3.7
`Bottom-Up`	83.6%	4
`Action-specific`	80.3%	3.8
`Actionness`	85.0%	3.4
`Random`	84.1%	4.4

Table 3.1: Baseline accuracies on the YouTube dataset. The reported results are averaged over all classes. The std is standard deviation over 25 folds.

each component of our system. Next, for a comparison with the state-of-the-art, we report results on both datasets.

3.4.2 Parameters selection

We first evaluate the impact of different trajectory lengths with our baseline `BoW` configuration on the YouTube dataset using the cross validation protocol. We noticed that up to a certain point, an increase in length L improves performance but afterwards since drifting from initial position is more probable to happen, it decreases slightly. The best performance is obtained with $L = 15$. Using lengths of 5 and 40, the accuracy is 3.8% and 0.5% lower respectively. We also evaluate the number of trajectory-groups K to use for the top-down segmentation with the `Actionnes` configuration. We obtain the best accuracy for $K = 20$. The algorithm is less sensitive to K and for $K = 5$ and $K = 50$ the acuracy is 0.7% and 1% lower respectively. More clusters split the video into too small regions that cannot be classified properly. Less clusters lead to a too coarse segmentation that again reduces the final performance. In the rest of the experiments we fix L to 15 and K to 20.

3.4.3 Results

Segmentation In this section we evaluate the impact of different segmentations on the recognition accuracy. As explained in section 3.3.1, a `BoW` based representation on trajectories is used, but this time we learn two different models, one for foreground and one for background.

In table 3.1 we report results for different segmentations on YouTube dataset.

In the first two rows we first evaluate how much a perfect segmentation (using ground truth test data, once at trajectory-level and once at trajectory-group level) can help to improve the final recognition. In case of trajectory-group level, the label of a trajectory is foreground if the majority of its points lie in the foreground bounding box. As expected, in both cases the obtained improvement over the `BoW` baseline is large (resp. 7% and 5%) and brings performance above the other methods. This proves that segmentation is crucial for good recognition. The difference between the two ground-truth (GT) accuracies is due to the coarse representation of our trajectory-groups that produces some quantization errors. As in all the following experiments we will be using the same trajectory-group representation, we already know that our best performance is probably bounded to 88.5%.

We first evaluate the `Bottom-Up` segmentation (see section 3.3.2). However, with this segmentation the improvement over a global bag-of-words is minimal. Next, the impact of top-down `Action-specific` and `Actionness` segmentation is reported. While the action-specific classifier performs poorly, the `Actionness` does a better job improving the baseline by 1.5%. Although the two models are driven by the same top-down principle, in the first case probably the classifier did not have enough data to perform a good and stable segmentation. Finally, we also report the accuracy for `Random` segmentation, where trajectory-groups are randomly assigned to background and foreground. Surprisingly this model performs better than BoW, bottom-up and action-specific. One possible explanation is that random segmentation does not depend on data, hence the trajectory-groups assigned to the two models (foreground and background) are uncorrelated While in case of a bad segmentation (`Bottom-Up`), at test time there can be a mismatch between segmentation and recognition and poor classification results are obtained.

Co-segmentation An orthogonal way to improve segmentation is co-segmentation (see section 3.3.3). We use the similarity among the foreground trajectory-groups from different videos of the same action to refine their segmentation. For all experiments, we set the ratio between unary and pairwise terms $\lambda = 0.2$ (see Eq.3.4).

In table 3.2 we report accuracy and the relative improvement produced by co-segmentation. At this point, the improvement brought by the co-segmentation seems to be limited or even negative. However, as we will see in the following sections, when the segmentation gets better, the impact of co-segmentation becomes more important.

Co-segmentation	Acc.	Impr.
`Action-specific`	77.0%	-3.3
`Actionness`	85.1%	+0.1
`Random`	83.6%	-0.5

Table 3.2: Mean accuracies for co-segmentation on the YouTube dataset. Improvements (Impr.) are relative to the corresponding segmentation of table 3.1.

Iterative Learning Next we evaluate the effect of the iterative learning (see section 3.3.4). In this experiment we can use different segmentations for the unary term in Eq.(3.7) as well as for the initialization for the iterative procedure (see table 3.3). For the unary term we test: `Itr.`, which is the iterative method using just the linear model defined in Eq.(3.6), `+Act.`, where we consider an additional unary term based on actionness, `+Co-seg.`, where we consider also co-segmentation, and `+Act.+Co-seg.`, where in conjunction with the iterative learning we use both actionness and co-segmentation. Also, as the optimization in this case is non-convex, the initialization of the segmentation is important for good results. We test three different initializations of the segmentations: random (`Rnd.`), based on actioness (`Act.`) and based on the ground-truth annotations (`Ann.`).

As action-specific always performs worse than `actionness` in previous tables, we did not report results for the former. As expected, best result is obtained when using the ground-truth annotations of the training data as initialization, although it is not far from those obtained by `actionness`. So `actionness` seems a good approximation of the ground truth segmentation. Looking at table 3.3, it is particularly interesting that the iterative learning with random segmentation for initialization and co-segmentation (denoted in the table as **(0)**) does not use ground-truth annotations on the training data but still is 2.2% better than the `BoW` baseline.

Non Linear Kernel An alternative way to improve results is by mapping the features into a kernel, such that non linear classifiers can be used. For this experiment we use the non linear kernel and settings defined in section 3.3.5. Unfortunately, we cannot combine these kernels with the iterative learning because the latter works only with linear kernels. We notice that when using kernels the normalization of the foreground and background descriptors highly affects the accuracy. Thus, in all the experiments that employ kernels we use a 5-fold cross-validation to yield the best normalization among ℓ_1, ℓ_2 and

Iterative Learning	Init.	Acc.	Impr.
Itr.	Ran.	85.0%	+1.5
+Act.	Rnd.	85.2%	+0.2
+Co-seg.(0)	Rnd.	85.7%	+2.2
+Act.+Co-seg.	Rnd.	85.7%	+0.6
Itr.	Act.	85.2%	+1.7
+Act.	Act.	86.1%	+1.1
+Co-seg.	Act.	86.2%	+2.7
+Act.+Co-seg.	Act.	86.7%	+1.6
Itr.	Ann.	85.5%	+2.0
+Act.	Ann.	86.4%	+1.4
+Co-seg.	Ann.	86.2%	+2.7
+Act.+Co-seg.(1)	Ann.	86.7%	+1.6

Table 3.3: Mean accuracies for iterative learning. Improvements (Impr.) are considered with respect to `Actionness` of table 3.1 for `+Act.`, with respect to `Actionness` of table 3.2 for `+Act.+Co-seg.` and with respect to `BoW` for the rest.

Kernel	Pool.	Acc.	Impr.
Random	llc	84.1%	+0.0
Random+Co-seg.	llc	83.9%	+0.3
Act.	llc	85.7%	+0.7
Act.+Co-seg.	llc	85.4%	+0.3
Random	hard	84.2%	+0.1
Random+Co-seg.	hard	84.2%	+0.6
Act.	hard	86.2%	+1.2
Act.+Co-seg.(2)	hard	86.8%	+1.7

Table 3.4: Mean accuracies for kernels. Improvements (Impr.) are relative to the corresponding configurations of tables 3.1 and 3.2.

unnormalized descriptors. We found that for the YouTube dataset, the best is ℓ_1 normalization, whereas for the UFC-sports dataset the best is to not normalize the descriptors. In table 3.4 we report the accuracy of different configurations with sparse coding+max pooling (`llc`) and with hard quantization+average pooling (`hard`).

3.4.4 Comparison with other methods

We compare the most promising configurations of our method with the state-of-the-art. We select configuration **(0)**, i.e. the iterative learning with co-segmentation, not using any kind of ground-truth information, configuration **(1)** which is also based on iterative learning but now initialized with ground truth annotations and combined with both `actionness` and co-segmentation and **(2)** which is based on `actionness` and co-segmentation mapped with a non linear kernel. Since each of **(0)**,**(1)** and **(2)** produce different segmentations, we finally report a weighted combination of the 3 classifiers in the last row of the table. The weights are selected by cross-validation on the training set.

Table 3.5 compares our results on the YouTube dataset. Improvements with respect to our baseline `BoW` range from +2.2 for configuration (0), which has the advantage of not using the ground-truth bounding boxes, to +3.3 for configuration **(2)**. This clearly shows that segmentation plays a relevant role for the final recognition accuracy and that there are several methods that can readily provide that useful segmentation. Comparing our methods with the state-of-the-art we observe that each one of the 3 methods is already comparable or better than most of the state-of-the-art excluding [Gaidon et al., 2011]. However, when combining our 3 segmentations we obtain a recognition accuracy comparable to [Gaidon et al., 2011].

Table 3.6 compares our results on the UFC-sports dataset. In this dataset the non linear kernel approach based on `actionness` and co-segmentation (2) seems better than the iterative learning. Also for this dataset the best result is obtained with the weighted combination of the 3 methods, which gives a final accuracy of 90.6 and 86.1 for the LOO evaluation and single split respectively. Figure 3.2 shows a more detailed, per class comparison between `Actionness`, the method of [Raptis et al., 2012] and the `BoW` baseline. Comparing our best

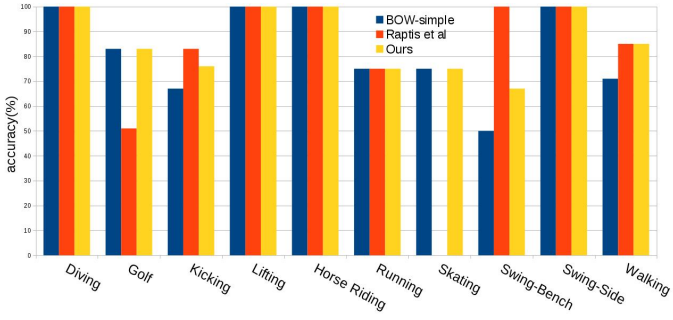

Figure 3.2: Per-class classification accuracy for UCF-Sports dataset.

result with the state-of-the-art, also in this dataset our accuracy is comparable with the best method [Todorovic, 2012] in LOO setting. Also for single split setting, we outperform most of the other methods including [Wang et al., 2011] which uses a simple bag of words on top of trajectories.

Method	Acc.	Impr.
[Brendel and Todorovic, 2010]	77.8%	-
[Wang et al., 2011]	84.2%	-
[Sapienza et al., 2012]	80%	-
[Gaidon et al., 2011]	87.9%	-
`(0)`	85.7%	+2.2
`(1)`	86.7%	+3.2
`(2)`	86.8%	+3.3
`(0)+(1)+(2)`	87.4%	+3.9

Table 3.5: Performance comparison on YouTube dataset with the state-of-the-art. Improvements (Impr.) are relative to baseline `BoW` without any segmentation that is reported in table 3.1

Method	Acc. LOO	Acc. Split
[Wang et al., 2011]	88.2%	-
[Kovashka and Grauman, 2010]	87.27%	-
[Todorovic, 2012]	92.1%	86.8%
[Lan et al., 2011]	-	73.1%
[Raptis et al., 2012]	-	79.4%
[Shapovalova et al., 2012]	-	75.3%
`(0)`	76.9%	80.1%
`(1)`	88.7%	81.5%
`(2)`	90.0%	86.1%
`(0)+(1)+(2)`	90.6%	86.1%

Table 3.6: Performance comparison on UCF-sports dataset with the state-of-the-art based on Mean per-class accuracies.

3.4.5 Qualitative Results

In Figure 3.3 we have shown some qualitative segmentation results of our method. As mentioned before, localization in videos are an ambiguous task in terms of quantitative measurements so we focused on the task of action

classification rather than segmentation, even though as shown in the Figure 3.3, the obtained segmentations obviously also provide us with some localization information.

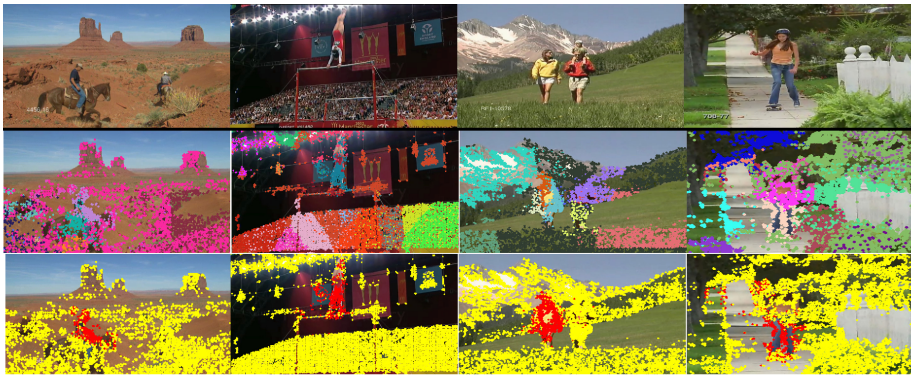

Figure 3.3: First row: A few frames from different action types. Second row: color-coded trajectory-groups. Each color represent a trajectory-group. Second row: after applying the Actionness. red dots belong to foreground while yellow points belong to background.

3.5 Conclusion

In this chapter we have shown that a good video segmentation is fundamental to obtain accurate action recognition. To this end we have proposed and evaluated several ways to integrate video segmentation and action recognition. In particular we have observed that coupling segmentation and recognition in an iterative learning can always improve the action recognition accuracy. Also, an alternative way to obtain similar results is to map the foreground and background description of the video into a non linear kernel. Finally, we have shown that joining different segmentations can further improve our results and reach state-of-the-art performance.

Moreover, we would like to mention some progress that has been achieved after this work. [Cai et al., 2014, Jain et al., 2013, Oneata et al., 2013] used more sophisticated representations beyond BoW representation to describe higher order statistics of features (like Fisher encoding or its variant VLAD) for recognizing actions. Recently, with increase in popularity of deep learning-based representations, many works utilize deep architectures to learn discriminative

models [Karpathy et al., 2014, Simonyan and Zisserman, 2014, Wang et al., 2015]. However, such deep models, unlike image recognition, can not significantly outperform previous hand-crafted representations so far. This can be due to the small amount of labeled data that is available for action recognition.

Finally, in Chapter 5, inspired by Actionness, we propose a method for the extraction of action proposals *i.e.* a set of temporal tubes from videos that most likely contain actions. Such action proposals can be used later for predicting the action label in a video and spatially localizing it.

Chapter 4

2D Representations for Viewpoint Estimation

In the previous chapter we investigated how disentangling foreground from background leads to a more effective representation for action recognition. However, so far we have employed traditional representations like BoW and LLC. In this chapter we leverage modern object representation methods to estimate the viewpoint of different kind of objects. This chapter addresses research question 2 by demonstrating the benefits of 2D modern representations for the fine-grained task of viewpoint estimation. Work corresponding to this chapter is published in:

- Ghodrati, A., Pedersoli, M. and Tuytelaars, T, *Is 2D Information Enough For Viewpoint Estimation?*, In British Machine Vision Conference (BMVC), 2014.

4.1 Introduction

Estimating the viewpoint of objects is a classical problem in vision. It aims at predicting a discrete or continuous viewpoint[1]. In conjunction with object

[1]Often in the computer vision community the term pose estimation and viewpoint classification are used interchangeably. However, to not confuse the classification of discrete or continuous viewpoints, as we do, with human pose estimation, that usually refers to articulated estimation of human body joints, here we will use the term viewpoint classification or estimation.

detection, viewpoint estimation is receiving increasing attention lately. Recent trends in the vision community suggest that, for an accurate estimation of the object viewpoint, 3D information about the object class is beneficial. For instance, Pepik *et al.* [Pepik et al., 2012b] show that using 3D CAD models of the class of interest can lead to a 3D representation of a deformable part model which, even though it has slightly worse detection performance, obtains state-of-the-art results in terms of viewpoint estimation. Likewise, Hejrati and Ramanan [Hejrati and Ramanan, 2012] show that providing 3D landmarks of cars can lead to a very accurate estimation of their 3D viewpoint.

However, 3D information (either 3D CAD models or 3D landmarks) is expensive to obtain and not available for many classes. In this chapter, we show that a very simple 2D architecture (in the sense that it does not make any assumption or reasoning about the 3D information of the object) generally used for object classification, if properly adapted to the specific task, can provide top performance also for viewpoint estimation.

More specifically in this work, we demonstrate on several datasets how a 1-vs-all classification framework based on a Fisher Vector (FV) pyramid and with neighbor viewpoints suppression (see sect. 4.3) can be used for viewpoint estimation. Furthermore, we investigate the performance of our system substituting the FV representation by the features extracted from a convolutional neural network (CNN) that recently has obtained very impressive results on the object classification task [Krizhevsky et al., 2012, Donahue et al., 2013, Razavian et al., 2014]. Our results show that for the fine-grained task of viewpoint estimation (up to 10°) both representations perform equally well and similarly or better than 3D methods previously proposed and designed specifically for the problem of viewpoint estimation.

This chapter is organized as follows. In section 4.2 we relate our method with the literature in 2D/3D viewpoint estimation. In section 4.3 we explain each component of our method and how those components interact with each other. Finally experiments are presented in section 4.4. Conclusions are drawn and future work is discussed in section 4.5.

4.2 Related Work

There are two lines of research for viewpoint estimation: one that uses 2D models for the viewpoint representation and the other that leverages on 3D information to tackle the problem. Inspired by the successes of deformable part models, several works have built 2D viewpoint-dependent detectors [Gu and Ren, 2010, Lopez-Sastre et al., 2011, Zhu and Ramanan, 2012]. Typically,

they explicitly handle viewpoints and discriminatively train models where the number of components corresponds to the views of an object for joint viewpoint classification and object detection. These models vary from rigid HOG templates [Ozuysal et al., 2009] to deformable part models [Gu and Ren, 2010, Lopez-Sastre et al., 2011, Zhu and Ramanan, 2012]. The drawback of such formulations is that they typically require training and evaluating a large number of components which can be computationally quite demanding. Recently Redondo-Cabrera *et al.* [Redondo-Cabrera et al., 2014] proposed a Hough Forest based method for simultaneous object detection and continuous viewpoint estimation. However their detection performance is not as good as DPM-based methods.

Latest progresses on viewpoint estimation have mostly utilized 3D CAD models [Stark et al., 2010, Pepik et al., 2012b, Pepik et al., 2012a, Liebelt and Schmid, 2012]. Pepik *et al.* [Pepik et al., 2012a] introduce a 3D extension of the deformable part model where part appearances as well as spatial deformations are represented in 3D. Such formulation allows synthesizing part appearance models for arbitrary viewpoints. Similarly, Zia *et al.* [Zia et al., 2013] first obtain a rough localization and viewpoint of the object by using an off-the-shelf method and then a continuous viewpoint is estimated by using annotated 3D CAD models. Arie-Nachimson and Basri [Arie-Nachimson and Basri, 2009] and Glasner *et al.* [Glasner et al., 2011] use viewpoint estimation as a prior in order to construct a 3D point cloud of object instances from training images. This limits their methods to datasets where such reconstruction is possible. Hejrati and Ramanan [Hejrati and Ramanan, 2012] estimate car viewpoints using an explicit 3D model of shape and viewpoint which is learned from structure-from-motion (SFM). The drawback of such methods is that they require labeled landmark positions of training data which is expensive to collect. Sun *et al.* [Sun et al., 2009] and Su *et al.* [Su et al., 2009] build 3D pose models by adopting the strategy of grouping local features into parts and learn part locations across viewpoints using generative models. Finally, Fanelli *et al.* [Fanelli et al., 2011] showed the usefulness of depth information for solving the problem of head pose estimation. They learn a mapping between simple depth features to 3D nose coordinates and rotation angles and estimate head pose through random forest based classifiers. In general, methods that rely on 3D models are not easy to collect for certain object categories.

Recently great interest has been expressed in Fisher kernel and convolutional neural network representations which have shown outstanding performance on several vision tasks. Simonyan *et al.* [Simonyan et al., 2013] showed that Fisher vectors on densely sampled SIFT features are capable of achieving state-of-the-art face verification performance. Toshev and Szegedy [Toshev and Szegedy, 2014] propose a cascade of deep neural network regressors that aim to

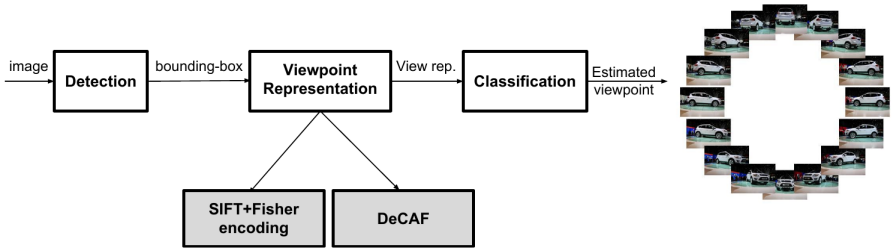

Figure 4.1: Our proposed pipeline for viewpoint estimation.

predict articulated human body joints. Jain *et al.* [Jain et al., 2014a] trained multiple convolutional neural nets to perform independent detection of parts. However, up to our knowledge, this is the first work that uses Fisher vector or convolutional neural network features in a simple 2D representation for the task of viewpoint estimation.

4.3 Proposed Pipeline

Our method takes as input a detection bounding box, extracts features and assigns to the bounding box a viewpoint. The estimation of the viewpoint is done with a one-vs-all classifier of a discrete set of viewpoints (see Fig. 4.1). In the rest of this section, we explain in detail each step of our pipeline.

4.3.1 Detecting the object of interest

For detection, we use a detector based on deformable part models (DPM) [Felzenszwalb et al., 2010, Girshick et al., 2010]. For each image, we apply the detector at multiple scales and collect detections which later are processed for estimating their viewpoints. For both training and testing, we train our model on the detected objects (i.e. we did not use ground-truth bounding-boxes for training). We empirically found out that using a detector for detecting object of interest during training is better than using ground-truth bounding-boxes of the object. One possible explanation is that training and test procedure need boxes that are generated from a same distribution (detector).

4.3.2 Feature extraction

We extract dense SIFT descriptors [Lowe, 2004] from the output of the detector. Specifically, we extract features over 5 scales, with a scaling factor of $\sqrt{2}$. 32×32 pixels patches are sampled with step size of 5 pixels from every detected bounding box. We call this basic feature representation `sift`. In addition, as proposed by Carreira *et al.* [Carreira et al., 2012], we also repeat the experiments with enriched SIFT descriptors where it is enriched with the location of the patch center with respect to the upper-left corner of the bounding box, normalized by its size. In this case, for each patch, the final descriptor is a $L1$-normalized concatenation of the SIFT descriptor and the patch location (`sift+loc`), resulting in a 130-dimensional descriptor.

4.3.3 Viewpoint representations

Fisher Vector FV [Jaakkola and Haussler, 1999] as explained in section 2.3 encodes information about the generative model that produces the low-level features by computing the gradient of the feature samples with respect to the parameters of the generative model. For computing the FV, we use the improved procedure proposed by Perronnin *et al.* [Perronnin et al., 2010] where a Gaussian Mixture Model (GMM) is fitted to dimensionality reduced SIFT features. Specifically, after reducing the feature dimensions to 60 using PCA, we estimate $\lambda_i = \{w_i, \mu_i, \Sigma_i, i = 1..K\}$, the parameters of the GMM, on a $100K$ sampled features set where w_i, μ_i, Σ_i are weight, mean, and diagonal covariance of the i-th mixture model respectively and K is the number of mixtures. Afterwards we estimate first and second order gradient statistics of each feature by computing its derivative w.r.t. the Gaussian means and variances. Let $G^X_{\lambda_i}$ be the weighted average of low-level features statistics with respect to component i,

$$G^X_{\lambda_i} = \frac{1}{N} \sum_{t=1}^{N} \alpha_i^t \psi(x_t; \lambda_i),$$

$$\psi(x_t; \lambda_i) = [\frac{1}{\sqrt{w}}(\frac{x_t - \mu_i}{\sigma_i}), \frac{1}{\sqrt{2w}}(\frac{(x_t - \mu_i)^2}{{\sigma_i}^2} - 1)].$$

Then each image is represented by stacking $G^X_{\lambda_i}$ for all mixtures: $[G^X_{\lambda_1}, ..., G^X_{\lambda_K}]$, where $X = \{x_t, t = 1..N\}$ is the set of N low-level features extracted from the image and α_i^t is the soft weight of the t-th feature for the i-th Gaussian. Following [Perronnin et al., 2010], we further normalise Fisher vectors by signed square-rooting and then $L2$ normalisation. In our experiments, we built

$K = 128$ Gaussian mixtures which leads to an image representation of length $2Kd = 2 \times 128 \times 60 = 15360$.

Also, it is known that in Fisher encoding, the spatial layout of the appearance is completely ignored (except when using the `sift+loc` enriched features). Without doubt, the spatial information may convey useful cues for viewpoint estimation so we encode spatial information in two ways. First, as a low-level strategy, by augmenting SIFT with location of the patch (as previously described) and second by building a Spatial Pyramid [Lazebnik et al., 2006] on top of the FV. In the experiments we use a spatial pyramid that divides the bounding box into 4×4, 2×2 and 1×1 cells and then stacks the FVs computed for each cell separately. We call this configuration `fisher+spm`. This leads in a final representation of length $15360 \times 21 = 322560$. As we will see in section 4.4.3, the two spatial encodings are complementary.

Convolutional Neural Networks We also experiment with the Deep Convolutional Activation Feature (`decaf`) [Donahue et al., 2013] (see section 2.5.1) to evaluate how good recently popular deep-learning approaches perform on viewpoint estimation. Decaf is based on the deep convolutional neural network architecture proposed by Krizhevsky *et al.* [Krizhevsky et al., 2012], which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [Deng et al., 2009]. In `decaf` the neurons activation of the late hidden layers of a pre-trained network are used as strong features for generic vision tasks with impressive results [Donahue et al., 2013]. We use the model pre-trained on ILSVRC since in viewpoint estimation there are too few training samples to properly learn a full deep representation from scratch. We do not perform any fine-tuning as well. The final network contains five convolutional layers followed by three fully-connected layers named by layer from 1 to 8. We refer to [Krizhevsky et al., 2012] for a detailed discussion of the architecture. For viewpoint training, we first extract the $L2$-normalized features from the pooled output of layer 5 (last convolutional layer). In this case, the viewpoint representation has length 9216.

4.3.4 Learning

We want to transform the discrete viewpoint estimation problem to a classification problem. To do so, we consider each viewpoint as a different class so the number of classes corresponds to the number of viewpoints. Then, for each viewpoint we learn a linear SVM based on a 1-vs-all strategy. In this scenario an important difference with a standard multi-class problem is that nearby viewpoints are generally visually very correlated. In this sense,

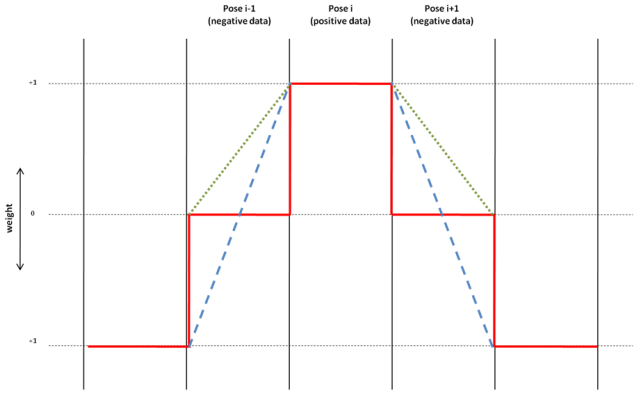

Figure 4.2: Different strategies for weighting nearby poses in negative samples (dotted, dashed and solid line). In this figure, pose i is considered as the positive class and pose i-1 and i+1 are nearby poses in the negative set. We found crisp weighting i.e. solid line performs best.

it is not reasonable to assign to all negative viewpoints equal importance. We tried different weighting strategies to reduce the impact of those negative samples that are close to the target viewpoint (figure 4.2) and imprically found out a hard ellimination is performing best. In the experimental results we show that eliminating the nearby viewpoints from the negative samples always improves the viewpoint estimation. We call this procedure "neighboring viewpoint suppression" or briefly `nv-suppression`. For very coarse binning, since it might happen that too much negative data is suppressed, therefore, whenever continuous viewpoint is provided, we suppress negative data only up to 10 degrees apart from the positive samples. Note that this is similar to the one-vs-most technique of Berg *et al.* [Berg et al., 2014].

4.4 Experiments

In this section we first describe the characteristics of the four datasets that we use and then on these datasets we thoroughly evaluate and compare several methods based on 2D and 3D information for estimating viewpoints.

4.4.1 Datasets for viewpoint estimation

Faces. We train the detector on a subset of the CMU MultiPIE face dataset [Gross et al., 2010] (see section 4.4.2) and test it on the Annotated faces-in-the-wild [Zhu and Ramanan, 2012] (AFW) dataset. Some examples of the AFW dataset are shown in Fig. 4.3. The CMU MultiPIE face dataset contains around 75000 images of 337 people over 13 viewpoints spanning over 180 degrees discretized every 15 degrees, with different illumination conditions and expressions. As in [Zhu and Ramanan, 2012], in our experiments, we use 900 faces for training with 300 of those frontal and the rest evenly distributed among other viewpoints. The AFW test set contains 468 faces from 205 images and 13 discretized viewpoints. Images contain cluttered backgrounds with large variations in face appearance. The metric used for this dataset is the same as in [Zhu and Ramanan, 2012] and reports the fraction of faces for which the estimated viewpoint is within some error tolerance (± 15 and ± 30). Notice that, to make the evaluation more realistic, missed detections are counted as errors in viewpoint estimation. In the following tables, we call this evaluation fraction of valid viewpoints (FVP).

Cars. We also present results on the EPFL Multi-view car dataset [Ozuysal et al., 2009] (Some examples are shown in Fig. 4.4). It contains 2299 images of 20 different car models. Cars are rotated over 360 degrees and their continuous viewpoint angle can be approximately calculated using the capture time of each image and the frontal view capturing time information that is provided. Images are captured using a static camera and all cars appear in the center of the image, without occlusions. Since a continuous viewpoint is provided for this dataset, in our evaluation we divide the viewpoints into 8, 16 and 36 discrete bins. We follow the experimental setup of [Ozuysal et al., 2009] and use the first 10 sequences for training and the next 10 sequences for testing. The evaluation metric that we use for this dataset is Mean Precision of Pose Estimation (MPPE) [Lopez-Sastre et al., 2011] and Median Angular Error (MAE). MPPE is computed as the average of the diagonal of the confusion matrix and MAE is the median error, where the error is computed as $min\{|\theta - \theta^*|, 360 - (|\theta - \theta^*|)\}$ with θ the estimated viewpoint angle and θ^* the ground truth viewpoint.

General Objects. Finally, we evaluate our methods on two general objects datasets: PASCAL3D+ [Xiang et al., 2014] and a subset of the 3DObject dataset [Savarese and Li, 2007]. PASCAL3D+ augments 12 rigid categories of the PASCAL VOC 2012 [Everingham et al., 2010] with 3D annotations including aeroplane, bicycle, boat, bottle, bus, car, chair, diningtable, motorbike, sofa, train and tvmonitor. Some examples of this dataset are shown in Fig. 4.5. For each category more images are added from ImageNet [Deng et al., 2009] and on average there are more than 3000 object instances per category. Since [Xiang

Figure 4.3: Example images from annotated faces-in-the-wild (AFW) testing set.

Figure 4.4: Example images from EPFL Multi-view car dataset.

Figure 4.5: Example images from 12 categories of the PASCAL3D+ dataset.

et al., 2014] reported the baseline using the `train` subset of PASCAL VOC 2012 (detection challenge) for training and `val` subset for evaluation, we follow the same protocol. For evaluation, we use the Average Viewpoint Precision (AVP). AVP takes into account the detection performance in the evaluation of the viewpoint estimation. In this way, an output from the detector is considered to be correct if and only if the bounding box overlap with the ground truth annotation is larger than 50% *and* the viewpoint is correct. As a result, AP is an upper bound for AVP. Note that Pose Estimation Average Precision (PEAP) proposed in [Lopez-Sastre et al., 2011] is different from AVP. PEAP[Lopez-Sastre et al., 2011] uses precision and recall of pose estimation whereas, even if not formally specified in [Xiang et al., 2014], AVP uses precision of pose estimation and recall of detection.

Finally, the 3DObject dataset [Savarese and Li, 2007] contains 10 everyday object classes such as iron, car and stapler. Each category includes 10 instances observed from 8 different viewpoints. Because other papers that use 3D information have published their results only on car and bicycle categories, we evaluate on car and bicycle as well. We follow the testing protocols of [Savarese and Li, 2007] and report results in terms of MPPE for this dataset.

4.4.2 Detection performance

In all experiments the first part of our algorithm consists of detecting the object of interest. For this we use standard DPM (`voc-release5`) [Girshick et al., 2010] with 6 components. For faces, we train DPM on 900 images of MultiPIE where the components are initialized based on the face orientation. We evaluate on AFW obtaining an AP of 88.3% with a maximum recall of 98.1%.

For detection on EPFL cars, we use the PASCAL VOC 2007 pre-trained DPM car model (`voc-release5`) [Girshick et al., 2010]. Its AP is 88.2% with a maximum recall of 100% on test images. For PASCAL3D+ dataset, following [Xiang et al., 2014], we train DPM (in this case we use version `voc-release4.01` to be compatible with the original results) on the `train` subset of PASCAL VOC 2012 and evaluate on `val`. The AP is reported in table 4.6 for each object category. For detection on Object3D cars and bicycles, we again use DPM car and bicycle models pre-trained on PASCALVOC-2007 (`voc-release5`) [Girshick et al., 2010]. Their APs on test data are 88.9% and 79.2% respectively with maximum recall of 100% and 96.8%.

		EPFL		AFW	
Feature Type	Encoding	MPPE (ground-truth)	MPPE (detector)	FVP±15 (ground-truth)	FVP±15 (detector)
sift	BoW	56.6%	54.8%	43.4%	49.4%
sift	fisher	68.4%	68.2%	51.1%	54.3%
sift	fisher+spm	82.1%	80.1%	73.3%	69.7%
sift+loc	fisher+spm	**82.8%**	**81.8%**	75.8%	**70.3%**
decaf	-	77.2%	72.0%	**77.3%**	67.9%

Table 4.1: An evaluation with training and testing data from ground-truth bounding boxes (3rd and 5th columns) and output of detector (4th and 6th columns) on the EPFL car dataset and AFW faces dataset. MPPE is computed as the average of the diagonal of the confusion matrix. FVP±15 is the fraction of faces that are within ±15 error interval, counting missed detections as infinite error.

4.4.3 Estimating viewpoint

In table 4.1, we evaluate the performance of different features and encoding on the EPFL car dataset using 8 view models, each covering 45 degrees and AFW using the 13 views learned on MultiPIE. We evaluate the methods either with `ground-truth` bounding boxes or with the bounding boxes obtained from a `detector` applied to all the images (both training and test). In general, as expected the ground truth bounding boxes give better results, but there are also some exceptions. `decaf` representation on ground-truth bounding boxes is among the best on both datasets, but when using the detector bounding boxes the performance surprisingly drops significantly. In contrast `fisher` is less sensitive to the bounding box localization. Considering the absolute performance of the different methods, we clearly notice that the baseline based on bag-of-words `BoW` (dictionary of size 4000 and the final representation is $L2$-normalized) is the poorest method for viewpoint representation. The best representation on both datasets is `fisher` with spatial pyramid `spm`. Comparing `sift` and `sift+loc`, we can see that also embedding spatial information in the low-level representation is still advantageous for viewpoint estimation. Also the performance of `decaf` is quite good, especially considering its much lower dimensionality. Based on these conclusions, we select the two last methods for the next experiments.

		EPFL			AFW	
Method	`nv-suppression`	8 bins	16 bins	36 bins	FVP±15	FVP±30
`fisher+spm`	×	**81.8%**	71.2%	46.4%	70.3%	84.2%
`fisher+spm`	✓	80.6%	**72.2%**	**51.8%**	**78.6%**	**90.6%**
`decaf`	×	72.0%	62.1%	39.1%	67.9%	82.3%
`decaf`	✓	**76.6%**	**67.8%**	**45.9%**	**86.5%**	**93.4%**

Table 4.2: The effect of suppressing negative neighboring viewpoints samples. On EPFL car dataset, MPPE is computed . Last two columns are fraction of faces that are within ±15 and ±30 error interval respectively, counting missed detections as infinite error for AFW dataset.

4.4.4 Effect of neighbor samples

In table 4.2 we investigate the impact of `nv-suppression` of negative samples explained in sect. 4.3.4. On EPFL, for the coarsest binning (8 bins), the suppression scheme does not help, probably because the confusion between nearby viewpoints in coarse binning is not an issue. However, when using a finer binning, the advantage of the `nv-suppression` is quite evident. Consequently, we continue our experiments with the suppression of the nearest neighbors enabled.

4.4.5 Layer selection in DeCAF

Next, we investigate the impact of features obtained from different layers of our convolutional neural network for the task of viewpoint estimation. To this end, we select the output of different layers as features and compute the performance for every layer as shown in figure 4.6. Note that we use 8 bins for the EPFL car dataset and apply the `nv-suppression` on negative samples on all `fisher`, `fisher+spm` and `decaf`. On the EPFL car dataset and the PASCAL3D+ dataset (table 4.6), `fisher+spm` outperforms `decaf` but for the AFW faces with 13 different viewpoints, `decaf` performs better. In addition, as it is shown, the last convolutional layer (layer 5) outperforms the others in both datasets. Finally, it is interesting to notice that the lower convolutional layers perform quite well whereas in other tasks generally they do not perform good. It makes sense in our case since `decaf` is optimized for the task of object classification and its deeper (fully conencted) layers are mostly dedicated for recognizing objects and therefore are robust to viewpoint variations. As mentioned before, we neither re-train nor fine-tune the network for the specific

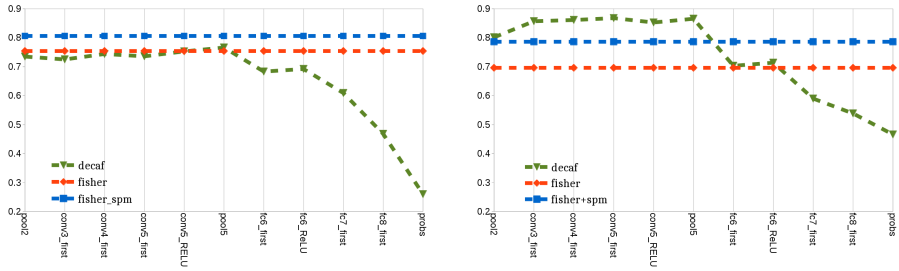

Figure 4.6: Evaluation of 8-bins viewpoint estimation problem on EPFL car dataset (left) and AFW face dataset (right) for different layers of the CNN network. Horizontal lines are Fisher vector performances.

task of viewpoint estimation and this explains the lower performance of deeper layers.

4.4.6 Computational cost

Another advantage of the proposed method is the reduced computational cost. Although a precise evaluation for each method in terms of time is difficult to obtain we can still reason about the computational cost of the different methods. We can safely claim that all the methods based on DPM are computationally more demanding than ours. This is due to the fact that in our method we use standard DPM models with 6 components and the following step, based on FV or Decaf has a negligible cost. For example, extracting SIFT, building a pyramid of Fisher vector and a 36-bins viewpoint classification takes on average 1.38, 0.73 and 0.19 seconds respectively on a quad-core processor using MATLAB on the EPFL dataset. Extracting decaf features for an image takes on average 0.2 seconds while the training time for 36 one-vs-all SVM linear models for `fisher+spm` and `decaf` is 290 and 6 seconds respectively. Instead, other methods generally use a DPM component for each view, so that, especially when searching for fine viewpoint estimation, the computational cost will be higher (e.g. detection using the standard DPM takes around 4 seconds for each EPFL image, while with 36 bins the computational cost of viewpoint-DPM should be 6 times the standard DPM model).

bins	[Ozuysal et al., 2009]	[Lopez-Sastre et al., 2011]	3D2PM-C Lin [Pepik et al., 2012a]	3D2PM-D [Pepik et al., 2012a]	ours (`fisher+spm`)	ours (`decaf`)
8 bins	-	73.7%	78.3%	78.5%	**80.6%**	76.6%
16 bins	41.6%	66.0%	69.0%	69.8%	**72.2%**	67.8%
36 bins	-	-	**52.1%**	45.8%	51.8%	45.9%

Table 4.3: Comparison with state of the art viewpoint classification methods on the EPFL dataset.

bins	3D2PM-C Lin [Pepik et al., 2012a]	3D2PM-D [Pepik et al., 2012a]	[Glasner et al., 2011]	ours (`fisher+spm`)	ours (`decaf`)
8 bins	**11.1**	12.9	24.8	12.5	13.5
16 bins	6.9	7.2	-	**6.75**	7.75
36 bins	**4.7**	5.8	-	5.0	6

Table 4.4: Viewpoint estimation in terms of MAE for EPFL car dataset (units are in degree and less is better).

4.4.7 Comparison with other methods

Table 4.3 compares our method to other state-of-the-art methods on the EPFL car dataset. `fisher+spm` outperforms all methods including 3D models on this dataset for 8 and 16 viewpoint bins and is slightly worse than the continuous model of [Pepik et al., 2012a] but outperforms their discrete version. `decaf` could not obtain state of the art performance on this dataset but it is on par with the discrete model of [Pepik et al., 2012a].

For the EPFL car dataset, as the angular viewpoint annotations are provided, we can also use the Median Angular Error for evaluation. Note that MAE is a metric for evaluation of continuous estimation but we are using a discrete estimation. Thus, for each bin, we assume the center of the bin as the estimated angular viewpoint. In terms of MAE, as shown in table 4.4, `fisher+spm` outperforms state-of-the-art discrete models (3D2PM-D) that use 3D CAD information and is on par with continuous appearance models (3D2PM-C Lin).

Table 4.5 shows the results of our methods and 3 other methods on the AFW dataset. Within ± 30 degree error tolerance, `fisher+spm` and `decaf` both perform well and outperform all the other methods whereas with ± 15 degrees error tolerance, `decaf` outperforms all other methods. These results are quite important especially considering that [Zhu and Ramanan, 2012] is tuned for face detection and pose estimation problems (they train part-based models where every facial landmark is considered as a part and use global mixtures to capture topological changes due to viewpoint) while `fisher+spm` and `decaf`

method	±15	±30
Face.com[1]	64.3%	86.5%
[Zhu and Ramanan, 2012]- indp. model	81.0%	89.0%
[Zhu and Ramanan, 2012]- shared. model	76.9%	87.0%
Multi-HoG[1]	74.6%	85.0%
ours(`fisher+spm`)	78.6%	90.6%
ours(`decaf`)	**86.5%**	**93.4%**

Table 4.5: Comparison with state of the art viewpoint estimation methods on the AFW face dataset.

are applicable to any other category.

For PASCAL3D+ dataset, the results of our methods and methods of [Xiang et al., 2014], [Pepik et al., 2012b] are shown in Table 4.6. As in [Xiang et al., 2014], we ignore the bottle category since its instances are often symmetric across different viewpoints. We notice the same trend as in the previous experiments: `fisher+spm` performs best on all viewpoint angles. `decaf` results are slightly lower but still comparable with [Pepik et al., 2012b] which relies on 3D CAD models. For more classes like train or sofa, our method performs markedly better than [Pepik et al., 2012b], whereas for other classes, like bicycle and car, [Pepik et al., 2012b] performs better. We believe this is correlated with the fact that for the latter classes, more and better 3D CAD models are available and therefore a better 3D representation can be learned.

[1]From Zhu *et al.* [Zhu and Ramanan, 2012]

AP/AVP	**aeroplane**	**bicycle**	**boat**	**bottle**	**bus**	**car**	**chair**	**diningtable**	**motorbike**	**sofa**	**train**	**tvmonitor**	**avg.**
[Xiang et al., 2014]-4V	40.0/34.6	45.2/41.7	3.0/1.5	-/-	49.3/26.1	37.2/20.2	11.1/6.8	7.2/3.1	33.0/30.4	6.8/5.1	26.4/10.7	35.9/34.7	26.8/19.5
[Pepik et al., 2012b]-4V	41.5/37.4	46.9/43.9	0.5/0.3	-/-	51.5/48.6	45.6/36.9	8.7/6.1	5.7/2.1	34.3/31.8	13.3/11.8	16.4/11.1	32.4/32.2	27.0/23.8
ours(`fisher+spm`)-4V	40.1/26.7	48.0/34.4	6.1/2.3	-/-	54.1/50.7	36.1/28.9	14.8/11.1	9.1/5.4	32.9/29.4	18.9/17.3	36.1/32.5	33.2/26.9	29.9/**24.1**
ours(`decaf`)-4V	40.1/24.5	48.0/32.9	6.1/2.4	-/-	54.1/49.6	36.1/24.1	14.8/10.7	9.1/6.1	32.9/27.6	18.9/14.2	36.1/32.2	33.2/27.6	29.9/22.9
[Xiang et al., 2014]-8V	39.8/23.4	47.3/36.5	5.8/1.0	-/-	50.2/35.5	37.3/23.5	11.4/5.8	10.2/3.6	36.6/25.1	16.0/12.5	28.7/10.9	36.3/27.4	29.9/18.7
[Pepik et al., 2012b]-8V	40.5/28.6	48.1/40.3	0.5/0.2	-/-	51.9/38.0	47.6/36.6	11.3/9.4	5.3/2.6	38.3/32.0	13.5/11.0	21.3/9.8	33.1/28.6	28.3/21.5
ours(`fisher+spm`)-8V	40.1/23.6	48.0/27.6	6.1/2.4	-/-	54.1/50.3	36.1/26.6	14.8/9.1	9.1/6.0	32.9/24.7	18.9/16.9	36.1/31.3	33.2/26.5	29.9/**22.3**
ours(`decaf`)-8V	40.1/17.7	48.0/27.7	6.1/1.9	-/-	54.1/49.6	36.1/23.3	14.8/7.8	9.1/4.8	32.9/27.1	18.9/11.1	36.1/31.2	33.2/26.4	29.9/20.8
[Xiang et al., 2014]-16V	43.6/15.4	46.5/18.4	6.2/0.5	-/-	54.6/46.9	36.6/18.1	12.8/6.0	7.6/2.2	38.5/16.1	16.2/10.0	31.5/22.1	35.6/16.3	30.0/15.6
[Pepik et al., 2012b]-16V	38.0/15.9	45.6/22.9	0.7/0.3	-/-	55.3/49.0	46.0/29.6	10.2/6.1	6.2/2.3	38.1/16.7	11.8/7.1	28.5/20.2	30.7/19.9	28.3/**17.3**
ours(`fisher+spm`)-16V	40.1/16.3	48.0/18.0	6.1/1.5	-/-	54.1/42.9	36.1/19.6	14.8/7.4	9.1/4.6	32.9/15.9	18.9/13.8	36.1/29.0	33.2/21.5	29.9/**17.3**
ours(`decaf`)-16V	40.1/11.2	48.0/19.5	6.1/1.7	-/-	54.1/43.5	36.1/19.4	14.8/6.3	9.1/4.6	32.9/20.6	18.9/10.5	36.1/28.6	33.2/22.3	29.9/17.1
[Xiang et al., 2014]-24V	42.2/8.0	44.4/14.3	6.0/0.3	-/-	53.7/39.2	36.3/13.7	12.6/4.4	11.1/3.6	35.5/10.1	17.0/8.2	32.6/20.0	33.6/11.2	29.5/12.1
[Pepik et al., 2012b]-24V	36.0/9.7	45.9/16.7	5.3/2.2	-/-	53.9/42.1	42.1/24.6	8.0/4.2	5.4/2.1	34.8/10.5	11.0/4.1	28.2/20.7	27.3/12.9	27.1/13.6
ours(`fisher+spm`)-24V	40.1/12.3	48.0/12.6	6.1/1.3	-/-	54.1/40.2	36.1/15.9	14.8/5.5	9.1/4.6	32.9/13.2	18.9/10.2	36.1/20.4	33.2/15.0	29.9/**13.7**
ours(`decaf`)-24V	40.1/10.0	48.0/12.9	6.1/0.8	-/-	54.1/39.8	36.1/16.7	14.8/4.9	9.1/4.5	32.9/13.4	18.9/7.4	36.1/21.7	33.2/18.9	29.9/**13.7**

Table 4.6: The results of [Xiang et al., 2014], [Pepik et al., 2012b] and ours for 4, 8, 16 and 24 viewpoint angles respectively on PASCAL3D+ dataset. The first number is AP of object detection and the second one is AVP of pose estimation.

category	[Payet and Todorovic, 2011]	[Lopez-Sastre et al., 2011]	[Pepik et al., 2012b]	[Glasner et al., 2011]	[Pepik et al., 2012a]	[Liebelt and Schmid, 2012]	ours (`fisher+spm`)	ours (`decaf`)
cars	86.1%	89.0%	**97.9%**	85.3%	95.8%	70.0%	95.8%	**97.9%**
bicycle	80.8%	90.0%	**98.9%**	-	96.0%	75.5%	98.1%	86.1%

Table 4.7: Viewpoint estimation on car and bicycle classes from Object3D dataset (MPPE).

For cars and bicycles of the 3D Object dataset for which objects are provided in 8 different viewpoints, as shown in table 4.7, both `decaf` and `fisher+spm` again outperform most of the other methods in the literature and achieve competitive performance to methods that use 3D CAD data ([Pepik et al., 2012b] and [Pepik et al., 2012a]).

4.5 Conclusion

In this chapter, we have presented a study of different methods for viewpoint estimation on four well-known and challenging datasets. Through an extensive evaluation we can clearly see that, in contrast to common believe, the very simple framework based on the extraction of features on the object bounding box using modern features (`decaf`) or in combination with modern encodings (`fisher+spm`) can in most of the cases outperform other methods in the litrature, including methods based on 3D or much more complex and computationally expensive models. This suggests that the next generation of viewpoint estimation methods should probably combine these powerful 2D representations with 3D reasoning.

Finally, same as last chapter, we would like to mention some progress that has been achieved after this work. Particularly, [Tulsiani and Malik, 2015] like us formulated viewpoint estimation as a classification task. They presented a Convolutional Neural Network based architecture and demonstrated significant improvements over state-of-the-art in viewpoint estimation. In another work, [Su et al., 2015] generated a large number of synthesized images of high variation using available 3D models which later were used to train a deep CNN for the viewpoint estimation task.

Chapter 5

Exploiting a Hierarchical Representation for Proposal Generation

In this chapter we start from a state-of-the-art feature learning method and exploit the different levels of information that it provides after the training procedure. This chapter directly addresses research question 3 by proposing a cascade that, going from high-level to low-level layers of a hierarchical representation, selects the most promising object/action locations. Work covered in this chapter is based on:

- Ghodrati, A.[1], Diba, A.[1], Pedersoli, M., Tuytelaars, T and Van Gool, L., *DeepProposal: Hunting objects by cascading deep convolutional layers*, In IEEE International Conference on Computer Vision (ICCV), 2015.

5.1 Introduction

In recent years, the paradigm of generating a reduced set of window candidates to be evaluated with a powerful classifier has become very popular in object detection. Indeed, most of the recent state-of-the-art detection methods [Cinbis

[1]indicates equal contribution: Amir worked more on implementation, experimental results and its extension to action proposals while Ali worked more on background overview and writing. They contributed equally for the rest including main idea and basic evaluations.

et al., 2013, Ren et al., 2015, He et al., 2015b, Wang et al., 2013] are based on such proposals. Furthermore, generating a limited number of proposals also helps weakly supervised object detection, which consists of learning to localize objects without any bounding box annotations [Deselaers et al., 2010, Song et al., 2014, Bilen et al., 2015].

Detection methods based on proposals can be seen as a two-stage cascade: first, a reduced set of promising and class-independent hypotheses, the *proposals*, are selected; and second, each hypothesis is classified in a class-specific manner. Similarly to sliding windows, this pipeline casts the detection problem to a classification problem. However, in contrast to sliding windows, more powerful and time consuming detectors can be employed at the class-specific stage, as the number of candidate windows is reduced.

Methods proposed in the literature for the generation of window candidates are based on two different approaches. The first approach uses bottom-up cues like image segmentation [Arbelaez et al., 2014, Van de Sande et al., 2011], object edges and contours [Zitnick and Dollár, 2014] for window generation. The second approach is based on top-down cues which learn to separate correct object hypotheses from other possible window locations [Alexe et al., 2010, Cheng et al., 2014]. So far, the latter strategy seems to have inferior performance. In this paper we show that, with the proper features, accurate and fast top-down window proposals can be generated.

We consider for this task the convolutional neural network (CNN) "feature maps" extracted from the intermediate layers of pretrained Alexnet-like [Krizhevsky et al., 2012] networks. We observe that classifiers trained on deeper convolutional layers, which form a more semantic representation, perform very well in recalling the objects with a reduced set of hypotheses. Unfortunately, as noticed also for other tasks [Hariharan et al., 2014], these layers provide a poor localization of the object due to their coarseness. In contrast, earlier layers are better in accurately localizing the object of interest, but their recall is reduced as they do not contain strong object cues. Here we propose a method for generating object proposals based itself on a cascade, starting from the last convolutional layer and going down with subsequent refinements to the initial layers of the net. As the flow of the cascade is inverse to the flow of the feature computation we call this an *inverse cascade*. Also, as we start from a coarse spatial window resolution, and throughout the layers we select and spatially refine the window hypotheses until we obtain a reduced and spatially well localized set of hypotheses, it is a *coarse-to-fine inverse cascade*. An overview of our approach, which we coined *DeepProposals*, is illustrated in Fig. 5.1.

More specifically, similarly to BING [Cheng et al., 2014], we select a reduced set of window sizes and aspect ratios and slide them on each possible location of the

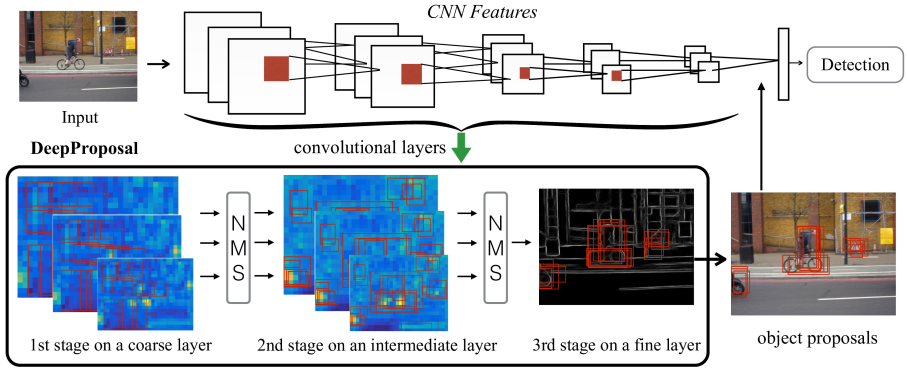

Figure 5.1: DeepProposals pipeline. Our method uses the activation layers of a deep convolutional neural network in a coarse-to-fine inverse cascading to obtain proposals for object detection. Starting from a dense proposal sampling in the last convolutional layer, we gradually filter out irrelevant boxes until reaching the initial layers of the net. After each stage, we apply Non-Maximum Suppression (NMS) to discard highly overlapped boxes. In the last stage we use contours extracted from an earlier layer with fine feature maps, to refine the proposals. Finally the generated boxes can be used within an object detection pipeline.

feature map generated by the last convolutional layer of a CNN. The relevance (or objectness) of the windows is evaluated using a linear classifier. As the proposal generation procedure should be fast, we base the feature aggregation for each candidate window on average pooling, which can be computed in constant time using integral images [Viola and Jones, 2004]. We filter out boxes that are less likely to contain an object and propagate the remaining ones to the next stage. In the second stage, we move to an earlier convolutional layer – in particular, the earliest layer that has the same feature map size as used in the first stage, and further filter the set of candidate boxes. As the number of boxes to evaluate has been reduced, we can afford adding more geometry in the representation by encoding each window with a pyramid representation. We re-rank the boxes and select the best N of them to pass to the last stage. Finally, in the third stage, we refine the localization obtained from the previous stage of the cascade, using an edgemap obtained from the second layer of the CNN.

We use the above framework not just for static images, but extend it to video, generating *action proposals*. To this end, we first apply the coarse-to-fine inverse cascade on each frame of a video. Then, we group the proposals into tubes,

by imposing time continuity constraints, based on the assumption that the object of interest has a limited speed. We show that such proposals can provide excellent results for action localization.

We evaluate the performance of the DeepProposals in terms of recall vs. number of proposals as well as in terms of recall vs. object (action) overlap. We show that in both evaluations the method is better than the current state of the art, and computationally very efficient. However, the biggest gains are achieved when using the method as part of a CNN-based detector like the Fast R-CNN proposed by [Girshick, 2015]. The features extracted from the generation of proposals will then be also the same features to be used for object detection. Thus, we can execute the full detection pipeline at a very low computational cost.

This paper is an extension of our earlier work [Ghodrati et al., 2015a]. In this version we include some additional related work; we apply the same framework to deeper network architectures; we evaluate it on more data sets and compare the method against some recent approaches. In addition, we extend DeepProposals for generating action proposals in videos and compare it with other state-of-the-art methods on two action datasets.

In the next section, we describe the most related work. Next, in section 5.4, we describe our proposed inverse coarse-to-fine cascade, followed by an detailed description of its components in section 5.5. In section 5.6, the extension of our method to action proposals is explained. Section 5.7 covers experiments for object proposals. It consists of an in-depth analysis of different components of the cascade and quantitative as well as qualitative comparison of our method with the state-of-the-art. In section 5.8, experiments for action proposal generation are described. Section 5.9 concludes the paper.

5.2 Related Work

Object proposal methods Object proposal generators aim at obtaining an accurate object localization with few object window hypotheses. These proposals can help object detection in two ways: searching objects in fewer locations to reduce the detector running time and/or using more sophisticated and expensive models to achieve better performance.

A first set of approaches measures the objectness of densely sampled windows (*i.e.* how likely is it for an image window to represent an object) [Alexe et al., 2010, Cheng et al., 2014, Zitnick and Dollár, 2014]. [Alexe et al., 2010] propose a measure based on image saliency and other cues like color and edges to

discriminate object windows from background. BING [Cheng et al., 2014] is a very fast proposal generator, obtained by training a classifier on edge features, but it suffers from low localization accuracy. Moreover, [Zhao et al., 2014] has shown that the BING classifier has minimal impact on locating objects and without looking at the actual image a similar performance can be obtained. Edgeboxes [Zitnick and Dollár, 2014] uses structural edges of [Dollár and Zitnick, 2013], a state-of-the-art contour detector, to compute proposal scores in a sliding window fashion without any parameter learning. For a better localization it uses a final window refinement step. Like these methods, our approach densely samples hypotheses in a sliding window fashion. However, in contrast to them, we use a hierarchy of high-to-low level features extracted from a deep CNN which has proven to be effective for object classification and detection [Girshick et al., 2014, Wang et al., 2013].

An alternative approach to sliding-window methods are the segmentation-based algorithms. This approach extracts from the image multiple levels of bottom-up segmentation and then merges the generated segments in order to generate object proposals [Arbelaez et al., 2014, Carreira and Sminchisescu, 2012, Manen et al., 2013, Van de Sande et al., 2011]. The first and most widely used segmentation-based algorithm is selective search [Van de Sande et al., 2011]. It hierarchically aggregates multiple segmentations in a bottom-up greedy manner without involving any learning procedure, but based on low level cues, such as color and texture. Multiscale Combinatorial Grouping (MCG) [Arbelaez et al., 2014] extracts multiscale segmentations and merges them by using the edge strength in order to generate object hypotheses. [Carreira and Sminchisescu, 2012] propose to segment the object of interest based on graph-cut. It produces segments from randomly generated seeds. As in selective search, each segment represents a proposal bounding box. Randomized Prim's [Manen et al., 2013] uses the same segmentation strategy as selective search. However, instead of merging the segments in a greedy manner it learns the probabilities for merging, and uses those to speed up the procedure. Geodesic object proposals [Krähenbühl and Koltun, 2014] are based on classifiers that place seeds for a geodesic distance transform on an over-segmented image.

Recently, following the great success of CNN in different computer vision tasks, CNN-based methods have been used to either generate proposals or directly regress the coordinates of the object bounding box. MultiBox [Erhan et al., 2014] proposes a network which directly regresses the coordinates of all object bounding boxes (without a sliding window fashion approach) and assigns a confidence score for each of them in the image. However, MultiBox is not translation invariant and it does not share features between the proposal and detection networks, i.e. it dedicates a network just for generating proposals. Its descendant, Single Shot Detector (SSD) [Liu et al., 2016], and another similar

work, [Redmon et al., 2016], eliminate proposals generation stage and directly detect objects which leads to fundamental improvement in speed. However, since they directly regress the bounding box locations, they are not still invariant to translation. DeepMask [Pinheiro et al., 2015] learns segmentation proposals by training a network to predict a class-agnostic mask (a mask that is independent of the class label) for each image patch and an associated score. Same as MultiBox, they do not share features between the proposal generation and detection. Moreover, they need segmentation annotations to train their network. OverFeat [Sermanet et al., 2013] is a method which does proposal generation and detection in one-stage. In OverFeat, region-wise features are extracted from a sliding window and are used to simultaneously determine the location and category of the objects. In contrast to it, our goal is to predict class-agnostic proposals which can be used in a second stage for class-specific detections.

Probably, the most similar to our work is the concurrent work of Region Proposal Network (RPN) proposed in [Ren et al., 2015]. RPN is a convolutional network that simultaneously predicts object bounds and objectness scores at each position. To generate region proposals, they slide a small network over the convolutional feature map. At each sliding-window location, they define k reference boxes (anchors) at different scales and aspect ratios and predict multiple region proposals parameterized relative to the anchors. Similarly to us, RPN builds on the convolutional features of the detection network. However, we leverage low-level features in early layers of the network to improve the localization quality of proposals. In addition, in contrast to them, our method can build on any pre-trained network without the need to re-train it explicitly for the proposal generation task.

Action proposal methods Action proposals are 3D boxes or temporal tubes extracted from videos that can be used for action localization, i.e. predicting the action label in a video and spatially localizing it. Also in this case, the main advantage of using proposals is to reduce the computational cost of the task and therefore make the method faster or allow for the use of more powerful classification approaches. The action proposal methods proposed in the literature to date mainly extend ideas originally developed for 2D object proposals in static images to 3D space. [Jain et al., 2014b] is an extension of selective search [Van de Sande et al., 2011] to video. It extracts super-voxels instead of super-pixels from a video and by hierarchical grouping it produces spatio-temporal tubes.

[Bergh et al., 2013] is an action proposal method inspired by the objectness method [Alexe et al., 2010], while a spatio-temporal variant of randomized Prime [Manen et al., 2013] is proposed in [Oneata et al., 2014]. Since most of

those methods are based on a super-pixel segmentation approach as a pre-processing step, they are computationally very expensive. To avoid such computationally demanding pre-processing, [van Gemert et al., 2015] proposed Action Localization Proposals (APT) which use the same features used in detection to generate action proposals.

Several action localization methods use 2D proposals in each frame but without generating intermediate action proposals at video-level. Typically, they leverage 2D object proposals that are generated separately for each frame in order to find the most probable path of bounding boxes across time for each action class separately [Gkioxari and Malik, 2015, Tran et al., 2014, Weinzaepfel et al., 2015, Yu and Yuan, 2015]. Our method is similar to these works in spirit. However, these methods use class-specific detectors for action localization while we propose a class-agnostic method to generate a reduced set of action proposals. The idea of using class-agnostic proposals allows us to filter out many negative tubes with a reduced computational time which enables the use of more powerful classifiers in the final stage.

5.3 CNN Layers for Proposals Generation

In this section we analyze the quality of the different layers of a CNN as features for window proposal generation. The window proposals ideally should cover all objects in an image. To evaluate the baselines in this section, we use the PASCAL VOC 2007 dataset [Everingham et al., 2010].

5.4 Overview of the Method

An overview of our inverse coarse-to-fine cascade is illustrated in Fig. 5.1. We first explain our method for static images, then later extend it to video in Section 5.6.

We start the search for object proposals in the top convolutional layer of the net. The feature maps at this layer have features well adapted to recognize high-level concepts like objects and actions, but have limited resolution. This coarseness leads to an efficient sliding window, yet results in poor localization results. We then gradually move to the lower layers, that use simpler features but have a much finer spatial representation of the image. As we go from a coarse to a fine representation of the image and we follow a flow that is exactly the opposite of how those features are computed we call this approach *coarse-to-fine inverse cascade*. We found that a cascade with 3 layers is an optimal trade-off between

complexity of the method and gain obtained from the cascading strategy. In the rest of this section we describe in detail the three stages of the cascade.

Stage 1: Dense Sliding Window on a Coarse Layer The first stage of the cascade uses the activation map of a one of the last convolutional layers of the network. This implies a high semantic representation, but also coarseness due to the multiple max pooling steps used in the network. As the feature representation is coarse, we can afford a dense sliding window approach with 50 different window sizes that best cover the training data in terms of size and aspect ratio. For details, see Sec. 5.5 (sliding window). The base descriptor of a given window is the sum pooling of the map activations that fall inside the window in the spatial dimension. We augment this descriptor with some information about the size and the aspect ratio of the window as detailed in Sec. 5.5 (bias on size and aspect ratio). The computation of the descriptor is carried out in a fast and size-independent way using an integral image representation. For details, see Sec. 5.5 (pooling). The scores of each window are the dot product of the window descriptor and the learned weights of a linear SVM classifier trained for discriminating between object and background, see Sec. 5.5 (classifier). We linearly re-scale the window scores to $[0, 1]$ such that the lowest and highest scores are mapped to 0 and 1 respectively. Afterwards we select the best $N_1 = 4000$ windows obtained from a non-maximum suppression algorithm (see Sec. 5.5 (non-maximum suppression)) before propagating them to the next stage.

Stage 2: Re-scoring Windows on an intermediate Layer At this stage, as we use a reduced set of windows, we can afford to spend more computation time per window. Therefore we add more geometry in the representation by encoding each window with a pyramid representation composed of two levels: 1×1 and 2×2, as described in Sec. 5.5 (pyramid). As in the first stage, we train a linear classifier with the aim to classify object versus background. The proposal scores from this layer are again mapped to $[0, 1]$. The final score for each proposal is obtained by multiplying the scores of both stages. Afterwards we apply a non-maximum suppression with overlap threshold $\beta + 0.05$ and select the 3000 best candidates. At the end of this stage, we aggregate the boxes from different scales using non-maximum suppression with threshold β and select the $N_{desired} = 1000$ best for refinement.

Stage 3: Local Refinement on a Fine Layer The main objective of this final stage is to refine the localization obtained from the previous stage of the cascade. For this stage we need higher resolution convolutional features in

order to grasp the low-level information which is suitable for the refinement task. Specifically, we refine the $N_{desired}$ windows received from the previous stage using the procedure explained in [Zitnick and Dollár, 2014] (see Sec. 5.5 (refinement)). To this end, we train a structured random forest [Dollár and Zitnick, 2013] on the second layer of the convolutional features to estimate contours similarly to DeepContour [Xinggang et al., 2015]. After computing the edge map, a greedy iterative search tries to maximize the score of a proposal over different locations and aspect ratios. It is worth mentioning that since our contour detector is based on the CNN-features, we again do not need to extract any extra features for this step.

5.5 Components of the Inverse Coarse-to-fine Cascade

Sliding window Computing all possible boxes in a feature map of size $N \times N$ is in the order of $O(N^4)$ and therefore computationally unfeasible. Hence, similarly to [Cheng et al., 2014], we select a set of window sizes that best cover the training data in terms of size and aspect ratio and use them in a sliding window fashion over the selected CNN layer. This approach is much faster than evaluating all possible windows and avoids to select windows with sizes or aspect ratios different from the training data and therefore probably false positives.

For the selection of the window sizes, we start with a pool of windows W in different sizes and aspect ratios $W : \{w | w \in \mathbb{Z}^2, \mathbb{Z} = [1..20]\}$. It is important to select a set of window sizes that gives high recall and at the same time produces well localized proposals. To this end, for each window size w, we compute its recall, R_α^w with different Intersection over Union (`IoU`) thresholds α and greedily pick one window size at a time. Specifically, we first compute recall of each window size w on multiple `IoU` of $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ on 300 randomly selected images from the training set and then select the best N window sizes using the method described in Algorithm 1. Using this procedure, $N = 50$ window sizes are selected for the sliding window procedure. In Fig. 5.4 (**middle**) we show the maximum recall that can be obtained with the selected window sizes, which is an upper bound of the achievable recall of our method.

Multiple scales Even though it is possible to cover all possible objects using a sliding window at a single scale of a feature map, it is inefficient since by using a single scale the stride is fixed and defined by the feature map resolution. For an efficient sliding window, the window stride should be proportional to the

Algorithm 1 Window size selection

Input: R_α^w recall of window size w over all objects at threshold α, and N number of best selected window sizes (set to 50), and W all possible window sizes.
Output: S^* best selected window sizes.
initialize: $S^* = \{\}$
$w^* \leftarrow \text{argmax}_w \sum_\alpha R_\alpha^w$
$S^* = \{w^*\}$
$W \leftarrow W \setminus \{w^*\}$
for $i = 2 \cdots N$ **do**
  for all $w_j \in W$ **do**
    $\mathbf{L}^{w_j} = \sum_\alpha R_\alpha^{S^* \cup w_j}$
  end for
  $w^* \leftarrow \text{argmax}_w \mathbf{L^w}$
  $S^* \leftarrow S^* \cup w^*$
  $W \leftarrow W \setminus \{w^*\}$
end for

window size. Therefore, in all the experiments we evaluate our set of windows at multiple scales. For each scale, we resize the image such that $min(w, h) = s$ where $s \in \{227, 300, 400, 600\}$. Note that the first scale is the network original input size.

Pooling As the approach should be very fast we represent a window by average pooling of the convolutional features that are inside the window. As averaging is a linear operation, after computing the integral image, the features of any proposal window can be extracted in a constant time. Let $f(x, y)$ be the specific channel of the feature map from a certain CNN layer and $F(x, y)$ its integral image. Then, average pooling Avg of a box defined by the top left corner $a = (a_x, a_y)$ and the bottom right corner $b = (b_x, b_y)$ is obtained as:

$$Avg(a, b) = \frac{F(b_x, b_y) - F(a_x, b_y) - F(b_x, a_y) + F(a_x, a_y)}{(b_x - a_x)(b_y - a_y)}. \tag{5.1}$$

Thus, after computing the integral image, the average pooling of any box is obtained in a constant time that corresponds to summing 4 integral values and dividing by the area of the box. We compute integral images for all feature maps of a particular layer, therefore the dimensionality of the feature vector of a window is equal to the number of channels (feature maps).

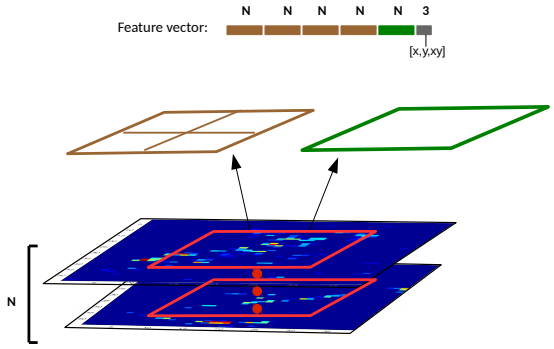

Figure 5.2: A spatial pyramid representation used for the second stage of our method.

Pyramid One of the main cues used to detect general objects is the object boundaries. Using an approach based on average pooling can dilute the importance of the object boundaries because it discards any geometrical information among features. Therefore, to introduce more geometry into the description of a window we consider a spatial pyramid representation [Lazebnik et al., 2006]. An illustration is shown in Figure 5.2. It consists of one full proposal window and a set of sub-windows. The sub-windows are generated by dividing the proposal window into a number of same size sub-windows (*e.g.* 2×2). Finally, each of them is represented and l_2 normalized separately.

Bias on size and aspect ratio Objects tend to appear at specific sizes and aspect ratios. Therefore, we add to the feature representation 3 additional elements: $(w, h, w \times h)$, where w and h are the width and height of a window. This can be considered as an explicit kernel which lets the SVM learn which object sizes can be covered at a specific scale. For the final descriptor, we normalize each pooled feature and size-related feature separately with the l_2 norm.

Classifier We train linear classifiers shared between all window sizes but for each scale and for each layer separately. For a specific scale and layer, we randomly select at most 10 windows per object that overlap the annotation bounding boxes more than 70%, as positive training data and 50 windows per image that overlap less than 30% with ground-truth objects as negative data. In all experiments we use a linear SVM [Fan et al., 2008] because of its simplicity and fast training. We did not test non-linear classifiers since they would be too slow for our approach.

Non-maximum suppression The ranked window proposals at each scale are finally reduced through a non-maximum suppression step. A window is removed if its IoU with a higher scored window is more than a threshold α, which defines the trade-off between recall and accurate localization. So, this threshold is directly related to the IoU criteria that is used for evaluation (see Sec. 5.7.2). By tuning α, it is possible to maximize the recall at an arbitrary IoU of β. Particularly, in this work we define two variants of our DeepProposals, namely DeepProposals50 and DeepProposals70, that maximize recall at IoU of $\beta = 0.5$ and $\beta = 0.7$ respectively. To this end, we fix α to $\beta + 0.05$, as suggested in [Zitnick and Dollár, 2014]. In addition, to aggregate boxes from different scales, we use another non-maximum suppression, fixing $\alpha = \beta$.

Refinement The refinement is based on an edge-based scoring function introduced in [Zitnick and Dollár, 2014]. The score is computed as difference between the contours wholly enclosed in a candidate bounding box and those that are not. A contour is wholly enclosed by a box if all edge pixels belonging to the contour lie within the interior of the box. Therefore, to compute the scoring function, we need to compute an edge response for each pixel in a given image. Edge responses are found by training 6 trees using a structured learning framework applied to random decision forests [Dollár and Zitnick, 2013]. As features, feature maps extracted from the fine layer of the CNN are fed to the structural random forest, considering each feature map as a channel for the algorithm. Given the edge map, the refinement is performed using a greedy iterative search to maximize the scoring function over position, scale and aspect ratio. At each iteration, the search step is reduced by half and the scores for new boxes are calculated. The search is halted once the translational step size is less than 2 pixels. Once the search is finished, the refined proposals are considered as our final proposals.

5.6 Proposals in Videos

Given a video sequence of length T, the goal is to generate a set of action proposals (tubes). Each proposal $P = \{R_1, ..., R_t, ..., R_T\}$ corresponds to a path from the box R_1 in the first frame to the box R_T in the last frame, and it spatially localizes the action (see Fig. 5.3). When the goal is to find proposals in videos, we need a) to capture the motion information that a video naturally provides, and b) to satisfy time continuity constraints.

One advantage of DeepProposals is that it can be put on top of any fine-to-coarse convolutional network regardless of its input/output (and possibly its

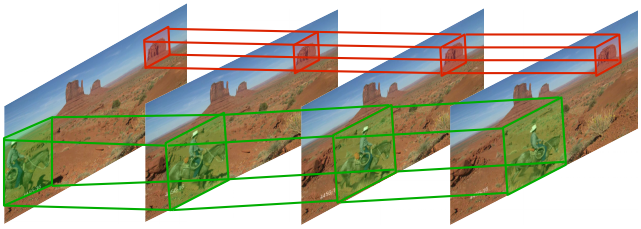

Figure 5.3: Two sample action proposals start from the first frame and end in the last frame. The green one is a correctly recalled action proposal while the red one is a false positive.

architecture). To benefit from both appearance and motion cues in a given video, we use two networks for the task of action proposal generation. The first network takes as input the RGB frames of a video, and is based on an Alexnet-like architecture, fine-tuned on VOC2007 for the task of object detection. The second network takes as input the optical flow of each frame extracted from the video. We use the motion-CNN network of [Gkioxari and Malik, 2015] trained on UCF101 (split1) [Soomro et al., 2012]. The architecture of this network is identical to that of the first network.

To generate a set of proposals in each frame, in the first and second stage of DeepProposals, we use an early fusion strategy, concatenating the feature maps generated by both networks and treating them as a single set of feature maps. For the last stage, since it is an alignment process, we only use the feature map of the appearance network.

So far the output is a set of proposals in each frame. In order to make the proposals temporally coherent, we follow the procedure of [Gkioxari and Malik, 2015] and link the proposals of each single frame over time into tubes. We define the linking scoring function between every two consecutive boxes R_t and R_{t+1} as follows:

$$S(R_t, R_{t+1}) = C(R_t) + C(R_{t+1}) + O(R_t, Rt + 1)$$

where $C(.)$ is the confidence score for a box and $O(.)$ is the intersection over union value if the overlap of the two boxes is more than 0.5, otherwise it is $-Inf$. Intuitively, the scoring function gives a high score if the two boxes R_t and R_{t+1} overlap significantly ($\mathbf{IoU}(box_1, box_2) \geq 0.5$) and if each of them most likely contains an object of an action.

Finally, we are interested in finding the optimal path over all frames. To this end, we first compute the overall score for each path P by $\sum_{t=1}^{T-1} \sum_{i,j \in P} S(R_t^i, R_{t+1}^j)$. Computing the score for all possible paths can be done efficiently using the

Stage	Layer	input candidates	Method	Pyramid	NMS	time
1	5	∼80.000	Slid. Window	1	Yes	0.30s
2	3	4.000	Re-scoring	$1 + 2 \times 2$	Yes	0.25s
3	2	1.000	Refinement	-	No	0.20s

Table 5.1: Characteristics of the stages of our inverse cascade (NMS: non maximum suppression).

Viterbi algorithm. The optimal path P^{opt} is then the one with the highest score. After finding the best path, all boxes in P^{opt} are removed and we solve the optimization again in order to find the second best path. This procedure is repeated until the last feasible path (those paths whose scores are higher than $-Inf$) is found. We consider each of these paths as an action proposal.

5.7 Experiments on Object Proposals

To evaluate our proposals, like previous works on object proposal generation, we focus on the well-known PASCAL VOC 2007 dataset. PASCAL VOC 2007 [Everingham et al., 2010] includes $9,963$ images divided in 20 object categories. $4,952$ images are used for testing, while the remaining ones are used for training.

We first evaluate the performance of each component of our approach and its influence in terms of recall and localization accuracy on Alexnet Architecture. We then compare the quality of our DeepProposals with multiple state-of-the-art methods. Detection results and run-time are reported for PASCAL VOC 2007 [Everingham et al., 2010], integrating DeepProposals in the Fast-RCNN framework [Girshick, 2015]. Finally, we evaluate the generalization performance of DeepProposals on unseen categories and some qualitative comparisons are presented.

5.7.1 Evaluation Metrics

We use two different evaluation metrics; the first is Detection Rate (or Recall) vs. Number of proposals. This measure indicates how many objects can be recalled for a certain number of proposals. We use Intersection over Union (`IoU`) as evaluation criterion for measuring the quality of an object proposal ω. `IoU` is defined as $|\frac{\omega \cap b}{\omega \cup b}|$ where b is the ground truth object bounding box. Initially, an object was considered correctly recalled if at least one generated window had an `IoU` of 0.5 with it, the same overlap used for evaluating the

detection performance of a method. Unfortunately, this measure is too loose because a detector, to work properly, also needs a good alignment with the object [Hosang et al., 2015]. Thus we evaluate our method for an overlap of 0.7 as well. We also evaluate recall vs. overlap for a fixed number of proposals. As shown in [Hosang et al., 2015], the average recall obtained from this curve seems highly correlated with the performance of an object detector built on top of these proposals.

5.7.2 Analysis of the Components

In this section, we investigate the effect of different parameters of our method, namely the different convolutional layers, the number of used window sizes and different levels of spatial pyramid pooling. We conduct this set of experiments without any cascading. Afterwards we investigate the effectiveness of different stages of DeepProposals.

Layers We evaluate each convolutional layer (from 1 to 5) of Alexnet [Krizhevsky et al., 2012] using the sliding window settings explained above. For the sake of simplicity, we do not add spatial pyramids on top of pooled features in this set of experiments. As shown in Fig. 5.4 (**left**) the top convolutional layers of the CNN perform better than the bottom ones. Also their computational cost is lower as their representation is coarser. Note this simple approach already performs on par or even better than the best proposal generator approaches from the literature. For instance, our approach at layer 3 for 100 proposals achieves a recall of 52%, whereas selective search [Van de Sande et al., 2011] obtains only 40%. This makes sense because the CNN features are specific for object classification and therefore can easily localize the object of interest.

However, this is only one side of the coin. If we compare the performance of the CNN layers for high overlap (see Fig. 5.4 (**middle**)), we see that segmentation-based methods like [Van de Sande et al., 2011] are much better. For instance the recall of selective search for 1000 proposals at 0.8 overlap is around 55% whereas ours at layer 3 is only 38%. This is due to the coarseness of the CNN feature maps that do not allow a precise bounding box alignment to the object. In contrast, lower levels of the net have a much finer resolution that can help to align better, but their encoding is not powerful enough to properly recall objects. In Fig. 5.4 (**middle**) we also show the maximum recall for different overlaps that a certain layer can attain with our selected sliding windows. In this case, the first layers of the net can recall many more objects with high overlap. This shows that a problem of the higher layers of the CNN is the lack of good spatial resolution.

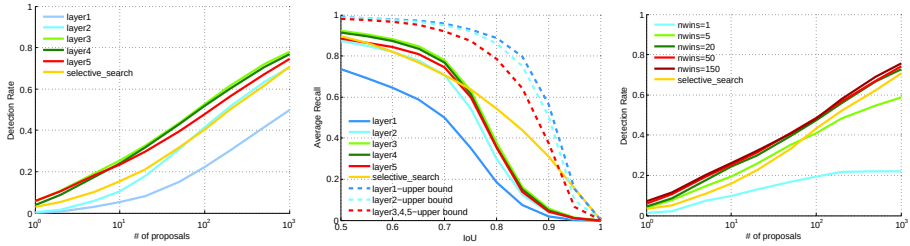

Figure 5.4: (**Left**) Recall versus number of proposals for `IoU`=0.7. (**Middle**) recall versus overlap for 1000 proposals for different layers of Alexnet. (**Right**) Recall versus number of proposals at `IoU`=0.7 on layer 5 for different number of window sizes. All are reported on the PASCAL VOC 2007 test set.

In this sense we could try to change the structure of the net in a way that the top layers still have high spatial resolution. However, this would be computationally expensive since applying convolutional filters are time consuming, and, more importantly, it would not allow to reuse the same features used for detection since there will be a mismatch between input image size of proposal generator and detector. Instead, in the next section we propose an efficient way to leverage the expressiveness of the top layers of the net together with the better spatial resolution of the bottom layers.

Number of Window Sizes In Fig. 5.4 (**right**) we present the effect of a varying number of window sizes in the sliding window procedure for proposal generation. The windows are selected based on the greedy algorithm explained in Sec 5.5. As the number of used window sizes increases, we obtain a better recall at the price of a higher cost. In the following experiments we will fix the number of windows to 50 because that is a good trade-off between speed and top performance. The values in the figure refer to layer 5 of Alexnet, however, similar behavior has been observed for the other layers as well.

Spatial Pyramid We evaluate the effect of using a spatial pyramid pooling in Fig. 5.5 (**left**). As expected, adding geometry improves the quality of the proposals. Moving from a pure average pooling representation (sp_level=0) to a 2×2 pyramid (sp_level=1) gives a gain that varies between 2 and 4 percent in terms of recall, depending on the number of proposals. Moving from the 2×2 pyramid to the 4×4 (sp_level=2) gives a slightly lower gain. At 4×4 the gain does not saturate yet. However, as we aim at a fast approach, we also need to consider the computational cost, which is linear in the number of

spatial bins used. Thus, the representation of a window with a 2×2 spatial pyramid is 5 times slower than a flat representation and the 4×4 pyramid is 21 times slower. For this reason, in our final representation we limit the use of the spatial pyramid to a 2×2 spatial pyramid.

Stages We now discuss the performance of the inverse cascade stage by stage in terms of both computational cost and performance. For Alexnet architecture we use layer numbers 14, 10, 6 for stage one to three respectively. A summary of the computational cost of each stage is given in Table 5.1. The entire cascade has a computational cost of 0.75 on a 8-core CPU of 3.50GHz, which is the composition of 0.3 , 0.25 and 0.2 for the first, second and third stage, respectively. Note that the first stage is very fast because even if we use a dense sliding window approach, with the integral image and without any pyramid level the cost of evaluating each window is very low.

As shown in Fig. 5.5 (**middle** and **right**), the second stage is complementary to the first and employed with a 2×2 pyramid improves the recall of the cascade by 5%. However, this boost is valid only up to an overlap of 0.75. After this point the contribution of the second stage is negligible. This is due to the coarse resolution of layer 5 and 3 that do not allow for a precise overlap of the candidate windows with the ground truth object bounding boxes. We found that, for our task, layer 3 and 4 have a very similar performance (Recall@1000 is 79% in both cases) and adding the latter in the pipeline did not help in improving performance (Recall@1000 is still 79%).

As shown in [Hosang et al., 2015], for a good detection performance, not only the recall is important, but also a good alignment of the candidates is needed. At stage 3 we improve the alignment without performing any further selection of windows; instead we refine the proposals generated by the previous stages by aligning them to the edges of the object. In our experiments for contour detection we observed that the first layer of the CNN did not provide as good a performance as layer 2 (0.61 vs. 0.72 AP on BSDS dataset [Arbelaez et al., 2011]), so we choose the second layer of the network for this task. Fig. 5.5 (**middle**) shows that this indeed improves the recall for high `IoU` values (above 0.7).

Network Architecture So far, we evaluated DeepProposals on a pre-trained Alexnet architecture trained on the Imagenet dataset. However, one advantage of DeepProposals is that it can be implemented on networks trained on different datasets or networks with different architectures. To this end, we setup our method on two other networks. The first is an alexnet-like architecture trained on the Places dataset [Zhou et al., 2014]. We used the pre-trained network of [Zhou et al., 2014] called placeNet for this purpose and use exactly the same

Layer	Feature map size	Recall(#1000,0.5)	Max(0.5)	Recall(#1000,0.8)	Max(0.8)
5	$36 \times 52 \times 256$	88%	97%	36%	70%
4	$36 \times 52 \times 256$	91%	97%	36%	79%
3	$36 \times 52 \times 256$	92%	97%	38%	79%
2	$73 \times 105 \times 396$	87%	98%	29%	86%
1	$146 \times 210 \times 96$	73%	99%	18%	89%

Table 5.2: Characteristics and performance of the CNN layers. Feature map size is reported for an image of size 600×860. Recall(#1000,β) is the recall of 1000 proposals for the overlap threshold β. Max(β) is the maximum recall for the overlap threshold β using our selected window sizes set.

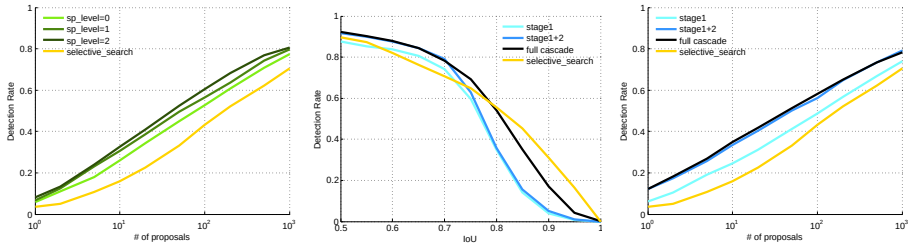

Figure 5.5: (**Left**) Recall versus number of proposals in `IoU`=0.7 for different spatial pyramid levels (**Middle**) Recall versus `IoU` for 1000 proposals for different stages of the cascade. (**Right**) Recall versus number of proposals in `IoU`=0.7 for the different stages of the cascade. All are reported on the PASCAL VOC 2007 test set.

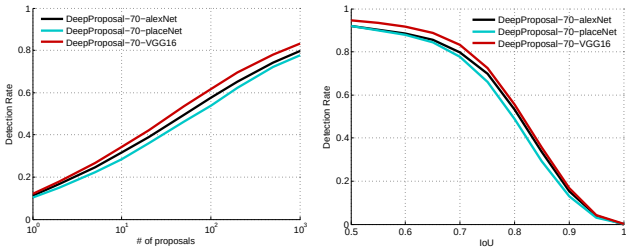

Figure 5.6: Recall versus number of proposals on the Pascal 2007 evaluation set for different network architectures with (**left**) IoU threshold of 0.7 and (**right**) Recall versus IoU threshold for 1000 proposal windows.

layer numbers as our original DeepProposals (i.e. layer numbers 14, 10, 6). The second architecture is the VGG-16 [Simonyan and Zisserman, 2015] network trained on Imagenet. This network is deeper compared to Alexnet and we use layers number 30, 24 and 12 for stage one to three, respectively. We did not change any other hyper-parameters of the method. Figure 5.6 shows the performance of DeepProposals on different networks. Surprisingly, placeNet works well particularly considering that it is trained to recognize scenes while we are using its feature maps for discovering objects. Its slightly lower performance compared to original DeepProposals can also be explained due to this fact. When the VGG-16 network is used we can observe a clear improvement in the quality of the generated proposals. This shows that with a more powerful network also the quality of the proposals is improved.

5.7.3 Comparison with state-of-the-art

In this section we compare the quality of the proposed DeepProposals with state-of-the-art object proposals on PASCAL 2007 [Everingham et al., 2010] and Microsoft COCO 2014 [Lin et al., 2014b] dataset. Fig. 5.7 show the recall with a varying number of object proposals or `IoU` threshold, for the PASCAL dataset. From Fig. 5.7 **top-left** and **top-right**, we can see that, even with a small number of windows, DeepProposals can achieve higher recall for any `IoU` threshold. Methods like BING [Cheng et al., 2014] and objectness [Alexe et al., 2010] provide a high recall only at $\texttt{IoU} = 0.5$, because they are tuned for `IoU` of 0.5.

In Table 5.3 we summarize the quality of the proposals generated by the most promising methods. Achieving 75% recall with `IoU` of 0.7 would be possible with 540 proposals of DeepProposals, 800 of Edge boxes, 922 of RPN-ZF, 1400 of selective search proposals and 3000 of Randomized Prim's proposals [Manen et al., 2013] on the PASCAL dataset.

Figure 5.7 **bottom-left** and **bottom-middle** show the curves related to recall over `IoU` with 100 and 1000 proposals. Again, DeepProposals obtain good results. The hand crafted segmentation based methods like selective search and MCG have good recall rate at higher `IoU` values. Instead DeepProposals perform better in the range of $\texttt{IoU} = [0.6, 0.8]$ which is desirable in practice and playing an important role in the object detectors performance [Hosang et al., 2015].

Figure 5.7 **bottom-right** shows average recall (AR) versus number of proposals for different methods. For a specific number of proposals, AR measures the proposal quality across `IoU` of [0.5, 1]. [Hosang et al., 2015] show that AR correlates well with detection performance. Using this criteria,

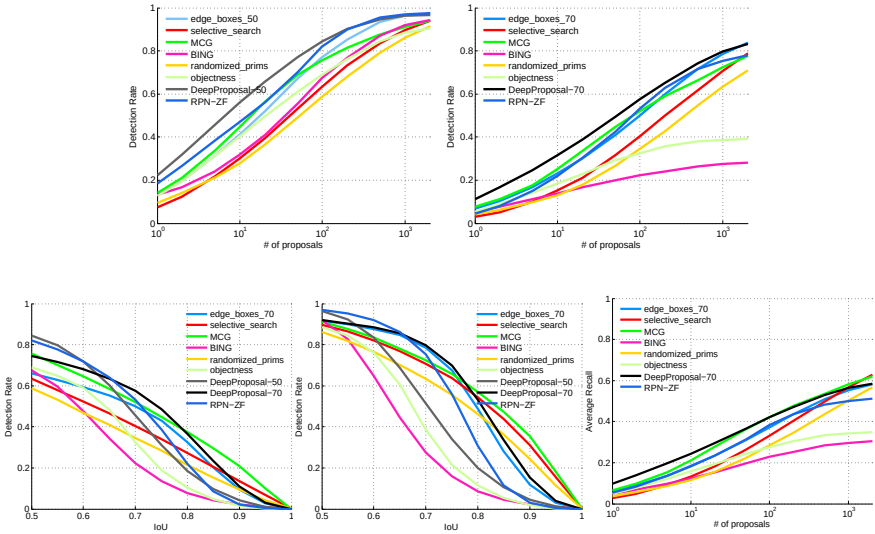

Figure 5.7: PASCAL VOC 2007 evaluation: Recall versus number of proposals for (**top-left**) IoU=0.5 and (**top-right**) IoU=0.7. Recall versus IoU threshold for (**bottom-left**) 100 proposals and (**bottom-middle**)1000 proposal . (**bottom-right**) Average Recall between IOU [0.5,1].

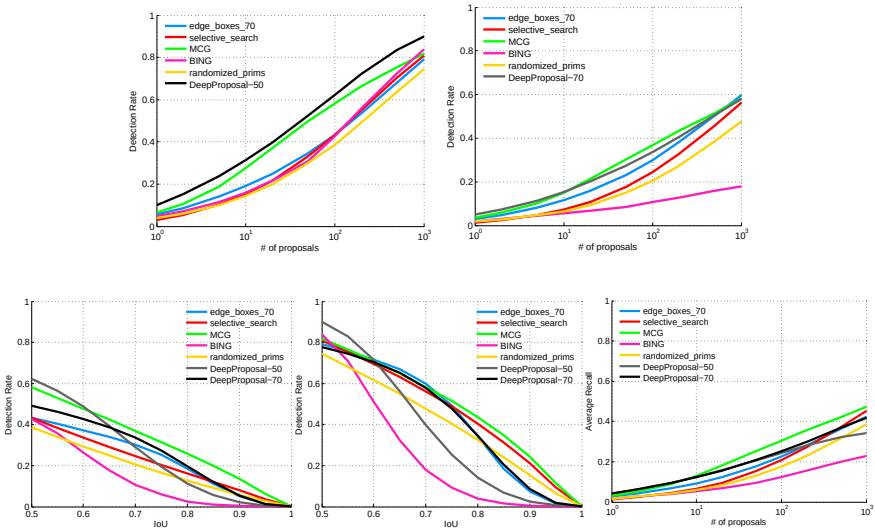

Figure 5.8:]
COCO 2014 evaluation: same as captions in Fig. 5.7.

DeepProposals are on par or better than other methods with 700 or fewer boxes. With more boxes, selective search and Edgeboxes perform better.

We also evaluate DeepProposals on COCO 2014 evaluation set. We train our svm classifiers on training set of COCO and evaluate our method on evaluation set. For COCO dataset, the areas less than 32×32 are considered as difficult and are ignored during evaluation. The same concept is also available for PASCAL but the definition of difficult objects is different than COCO. However, our method is not designed to localize small objects since the window sizes we chose is tuned for PASCAL. In Fig. 5.8 **top-left** and **top-right** we show the recall with a varying number of object proposals. Fig. 5.8 **bottom-left** and **bottom-middle** show the curves related to recall over `IoU` with 100 and 1000 proposals respectively and Fig. 5.8 **bottom-right** shows average recall(AR) versus number of proposals. In general, the trend is same as the trend on PASCAL dataset except that MCG performs better in most of the cases.

Run-time The run-time tests for our proposed method and the others are also available in Table 5.3. Since our approach uses the same CNN features used by state-of-the-art object detectors like RCNN [Girshick et al., 2014] and SppNet [He et al., 2015b], it does not need any extra cues and features and we can consider just the running time of our algorithm without the CNN extraction time [2]. DeepProposals takes 0.75 seconds on CPU and 0.4 seconds to generate object proposals on a GeForce GTX 750 Ti GPU, which is slightly slower than Edgeboxes. The fastest method is RPN-ZF, a convolutional network based on [Zeiler and Fergus, 2014] network architecture, tuned for generating object proposals. Note that for RPN-ZF, the running-time on a GPU is reported while the others are reported on a CPU. The remaining methods are segmentation based and take considerably more time.

Qualitative Results Figure 5.9 and 5.10 show some qualitative results of DeepProposals and another state of the art method, Edge boxes. In general, when the image contains high-level concepts cluttered with many edges our method gives better results (Figure 5.9). However, for small objects with clear boundaries edge boxes performs better (Figure 5.10) since it is completely based on contours and can easily detect smaller objects. In Figure 5.11 we also show a map indicating the location of first 10 object proposals. It illustrates how our method focuses on the objects even with a few proposals while EdgeBoxes mainly spots the close-shaped regions of the image.

[2] If CNN features have to be (re)computed, that would add 0.15 sec. extra computation time on our GPU.

Figure 5.9: Qualitative examples of our object proposals (1st and 3rd column) versus Edge boxes proposals (2nd and 4th column). In these examples our method is performing better than the state-of-the-art EdgeBoxes method. An object is correctly localized if its `IoU` with the ground-truth bounding box is more than 0.7. We use **1000** proposals per image for each method. Blue boxes are the closest proposal to each ground truth bounding box. Red and green boxes are ground-truth boxes where green indicates a localized object while red indicates a missed object.

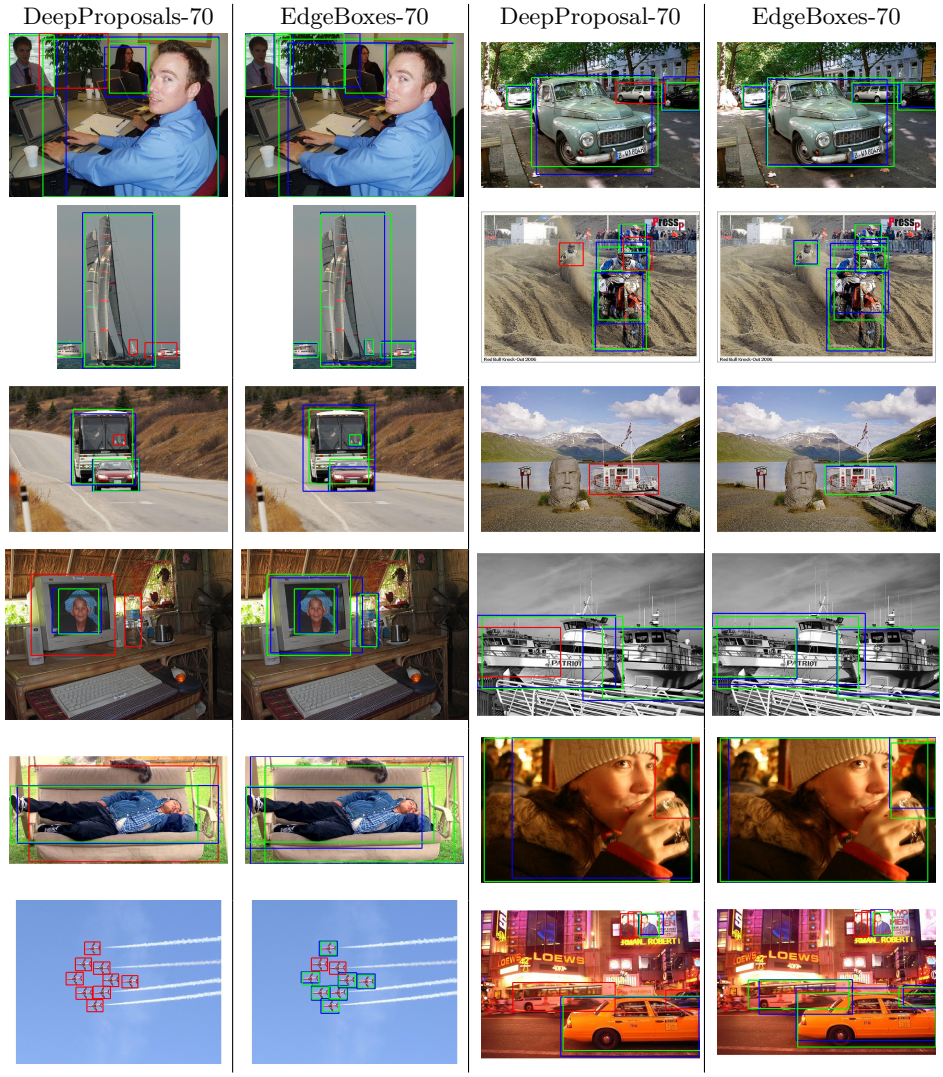

Figure 5.10: Qualitative examples of our object proposals (1st and 3rd column) versus Edge boxes proposals (2nd and 4th column). In these examples Edgeboxes is performing better than ours. An object is correctly localized if its `IoU` with the ground-truth bounding box is more than 0.7. We use **1000** proposals per image for each method. Blue boxes are the closest proposal to each ground truth bounding box. Red and green boxes are ground-truth boxes where green indicates a localized object while red indicates a missed object.

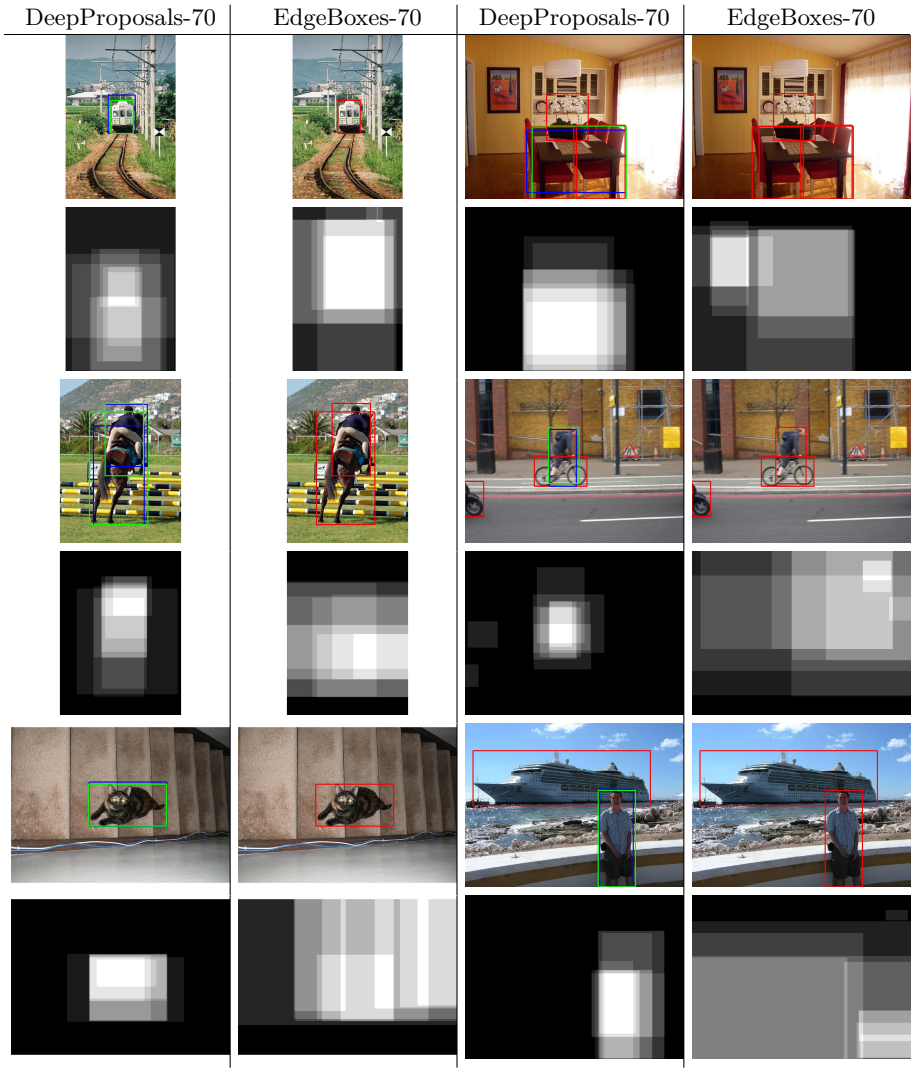

Figure 5.11: Qualitative examples of our object proposals (1st and 3rd column) versus edge boxes proposals (2nd and 4th column). An object is correctly localized if its IoU with ground-truth bounding box is more than 0.7 using **10** proposals for each method. For each image, we also show a map indicating the location of first 10 object proposals. It illustrates how our method focuses on the objects even with a few proposals while EdgeBoxes mainly spots the close-shaped regions of the image.

	AUC	N@25%	N@50%	N@75%	Recall	Time
BING [Cheng et al., 2014]	.19	292	-	-	29%	.2s
Objectness [Alexe et al., 2010]	.26	27	-	-	39%	3s
Rand. Prim's [Manen et al., 2013]	.30	42	349	3023	71%	1s
Selective Search [Van de Sande et al., 2011]	.34	28	199	1434	79%	10s
Edge boxes 70 [Zitnick and Dollár, 2014]	.42	12	108	800	**84%**	.3s
MCG [Arbelaez et al., 2014]	.42	9	81	1363	78%	30s
RPN-ZF [Ren et al., 2015]	.42	13	83	922	78%	**.1s***
DeepProposals70	**.48**	**5**	**53**	**540**	83%	.75s

Table 5.3: Our method compared to other methods for `IoU` threshold of 0.7. AUC is the area under recall vs. `IoU` curve for 1000 proposals. N@25%, N@50%, N@75% are the number of proposals needed to achieve a recall of 25%, 50% and 75% respectively. For reporting Recall, at most 2000 boxes are used. The run-times for the other methods are obtained from [Hosang et al., 2015]. *In contrast to the other methods, for RPN-ZF the run-time is evaluated on GPU.

Object Detection Performance In the previous experiments we evaluated our proposal generator with different metrics and showed that it is among the best methods for all of them. However, we believe that the best way to evaluate the usefulness of the generated proposals is a direct evaluation of the detector performance. Indeed, recently it has become clear (see [Hosang et al., 2015]) that an object proposal method with high recall at 0.5 IoU does not automatically lead to a good detector.

Some state-of-the-art detectors at the moment are: RCNN [Girshick et al., 2014], SppNet [He et al., 2015b], fast-RCNN [Girshick, 2015]. All are based on CNN features and use object proposals to localize the object of interest. RCNN uses the window proposals to crop the corresponding regions of the image, compute the CNN features and obtain a classification score for each region. This approach is slow and takes around 10 sec on a high-end GPU and more than 50 sec on the GPU used for our experiments (GeForce GTX 750 Ti). SppNet and fast-RCNN instead compute the CNN features only once, on the entire image. Then, the proposals are used to select the sub-regions of the feature maps from where to pull the features. This allows this approach to be much faster. With these approaches then, we can also reuse the CNN features needed for the generation of the proposals so that the complete detection pipeline can be executed without any pre-computed component, roughly in 1 second on our GPU (GeForce GTX 750 Ti).

Concurrently to our method also the Faster-RCNN was recently introduced [Ren et al., 2015]. It uses a Region Proposal Network (RPN) for generating proposals that shares full-image convolutional features with the detection network.

We compare the detection performance of our DeepProposals70 with selective search and RPN proposals. For RPN proposals the detector is trained as in the original paper [Ren et al., 2015] with an alternating procedure, where detector and localization sub-network update the shared parameters alternatively. Our method and selective search are instead evaluated using a detector fine-tuned with the corresponding proposals, but without any alternating procedure, *i.e.* the boxes remain the same for the entire training. The training is conducted using the faster-RCNN code on PASCAL VOC 2007 with 2000 proposals per image. In Fig. 5.12 we report the detector mean average precision on the PASCAL VOC 2007 test data for different number of used proposals.

The difference of selective search with CNN-based approaches is quite significant and it appears mostly in a regime with low number of proposals. For instance, when using 50 proposals selective search obtains a mean average precision (mAP) of 28.1, while RPN and our method obtain a mAP already superior to 50. We believe that this difference in behavior is due to the fact that our method and RPN are supervised to select good object candidates, whereas selective search is not.

Comparing our proposals with RPN, we observe a similar trend. DeepProposals produces superior results with a reduced amount of proposals (< 100), while RPN performs better in the range of between 100 and 700 proposals. With more than 700 proposals both methods perform again similarly and better than selective search. Finally, with 2000 proposals per image, selective search, RPN and DeepProposals reach the detection performance of 59.3, 59.4 and 59.8, respectively.

Thus, from these results we can see that RPN and our approach perform very similarly. The main difference between the two approaches lies in the way they are trained. Our approach assumes an already pre-trained network, and learns to localize the object of interest by leveraging the convolutional activations generated for detection. RPN instead needs to be trained together with the detector with an alternating approach. In this sense, our approach is more flexible because it can be applied to any CNN based detector without modifying its training procedure.

5.7.4 Generalization to Unseen Categories

We evaluate the generalization capability of our approach on the Microsoft COCO dataset [Lin et al., 2014b]. The evaluation of the approach has been done by learning either from the 20 classes from VOC07 or COCO or from 1, 5, 20, 40, or 80 categories randomly sampled from COCO. As shown in figure 5.13, when the DeepProposals are trained using only 5 classes, the recall

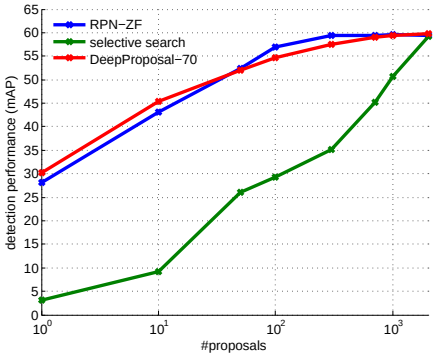

Figure 5.12: Detection results on PASCAL VOC 2007.

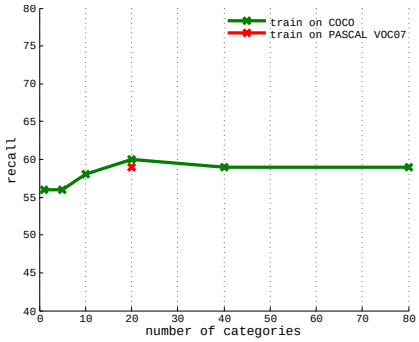

Figure 5.13: Generalization of DeepProposals: we train models with different number of categories and evaluate it on the whole eval-set of COCO dataset. We set IoU threshold to 0.5 and number of proposals to 1000.

at 0.5 IoU with 1000 proposals is slightly reduced (56%). With more classes, either using VOC07 or COCO, recall remains stable around 59% - 60%. This shows that the method can generalize well over all classes. We believe this is due to the simplicity of the classifier (average pooling on CNN features) that avoids over-fitting to specific classes. Note that in this case our recall is slightly lower than the Selective Search with 1000 proposals (63%). This is probably due to the presence of very small objects in the COCO dataset, that are missed by our method as it was not tuned for this setting. These results on COCO demonstrate that our proposed method is capable to generalize learnt objectness beyond the training categories.

5.8 Experiments on Action Proposals

5.8.1 Evaluation

We evaluate the performance of the inverse cascade for action proposals on the UCF-Sports [Rodriguez et al., 2008] dataset. We train our models on a training set that contains a total of 6,604 frames. Additionally, we have 2,976 frames in the test set, spread over 47 videos.

Like [van Gemert et al., 2015], we measure the overlap between an action proposal and the ground-truth video tubes using the average intersection-over-union score of 2D boxes for all frames where there is either a ground-truth box or proposal box. Formally:

$$Ovr(P, G) = \frac{1}{|F|} \sum_{t \in F} \frac{P_t \cap G_t}{P_t \cup G_t},$$

where P and G are action proposal and ground-truth action tube, respectively. F is the set of frames where either P or G is not empty. G_t is empty if there is no action in the frame t of the ground-truth tube. In this case, $P_t \cap G_t$ is set to 0 for that frame.

Considering each frame as an image and applying DeepProposals on each frame individually, the frame-based recall of objects/actors for an `IoU` of 0.7 is 78% for 10 windows and 96% for 100 windows. One possible explanation for such a promising frame-based recall is that an action mainly contains an actor performing it and hunting that actor in each frame is relatively easier than hunting general objects in the task of object proposal generation. However, this does not take into account any temporal continuity. Constructing tubes from these static windows, which results in our action proposals, is our final goal.

The extension of the inverse cascade for actions introduces an additional parameter which is the number of windows that we select in each frame. In figure 5.14 (**left**) we show the recall of the action proposals while varying the number of windows we select per frame. As expected, selecting more windows in each frame leads to a higher recall of the action proposals. However, it also leads to an increasing computational cost, since the computational complexity of the Viterbi algorithm is proportional to the square of the number of windows per frame. For example, the Viterbi algorithm for a video of length 240 frames takes 1.3 and 12.1 seconds for $N = 100$ and $N = 300$ respectively. During all the following experiments we select $N = 100$ windows per frame to have a good balance between performance and time complexity.

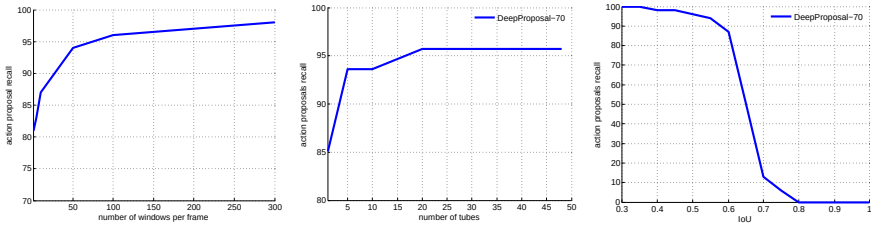

Figure 5.14: (**left**) Action proposals recall at IoU of 0.5 varying the number of windows per frame. (**middle**) Recall versus number of proposals in `IoU`=0.5 for the different number of action proposals. (**right**) Action proposal recall at different IoU values for 20 action proposals. All results are reported on the UCF-sports test set.

Figure 5.14 (**middle**) shows the action proposals recall for different number of proposals. As it is shown even for a very small number of proposals, DeepProposals obtains very good performance as already observed also for object proposals. Figure 5.14 (**right**) shows the recall of our method for 20 action proposals (tubes) per video over different IoU values. Our method works very well in the regime of [0.3..0.6]. Notice that the definition of action proposals recall is different than object proposals recall and the performance in `IoU`=0.5 is already quite promising.

5.8.2 Comparison with the state of the art

We evaluate our action proposals on two different datasets namely UCF-Sports [Rodriguez et al., 2008] and UCF101 [Soomro et al., 2012]. UCF-Sports contains 10 action categories and consists of 150 video samples, extracted from sport broadcasts. The actions in this dataset are temporally trimmed. For this dataset we use the train and test split proposed in [Lan et al., 2011]. UCF101 is collected from YouTube and has 101 action categories where 24 of the annotated classes (corresponding to $3,204$ videos) are used in literature for action localization. In this dataset, for evaluation we report the average recall of 3 splits. Finally, for both datasets, we select the first top 100 boxes in each frame and find the N best paths over time for each video.

In Table 5.4 we compare our proposal generation method against state-of-the-art methods in the presented datasets. As shown, our method is competitive or improves over all other methods with fewer proposals. In the UCF-Sports dataset, DeepProposals have higher recall compared to the recently published

	Recall	#proposals
UCF-Sports		
[Brox and Malik, 2010]	17.02	**4**
[Jain et al., 2014b]	78.72	1642
on average [Oneata et al., 2014]	68.09	3000
[Gkioxari and Malik, 2015]	87.23	100
APT [van Gemert et al., 2015]	89.36	1449
DeepProposals	**95.7**	20
UCF101		
APT [van Gemert et al., 2015]	37.0	2304
DeepProposals	**38.6**	**34**

Table 5.4: Our action proposals generator compared to other methods at a `IoU` threshold of 0.5. The number of proposals is averaged over all test videos. All the reported numbers on UCF-sports except ours are obtained from [van Gemert et al., 2015]. For UCF101, like ours, we report the APT performance for split3.

APT proposal generator [van Gemert et al., 2015] with almost 70x fewer proposals. Notice that the method proposed by [Brox and Malik, 2010] is designed for motion segmentation and we use it here to emphasize the difficulty of generating good video proposals. In the UCF101 dataset we see the same trend, we outperform APT while using 67× fewer proposals.

Run-time Computationally, given the optical flow images, our method needs 1.2 seconds per frame to generate object proposals and on average 1.3 seconds for linking all the windows. Most of the other methods are an order of magnitude more expensive mainly because of performing super-pixel segmentation and grouping.

Qualitative Results In figure 5.15 we provide examples of our action proposals extracted from some videos of UCF-sports dataset. For each video we show 5 cross sections of a tube. These sections are equally distributed in the video.

5.9 Conclusion

DeepProposalsis a new method to produce fast proposals for object detection and action localization. In this paper, we have presented how DeepProposals produces proposals at a low computational cost through the use of an efficient

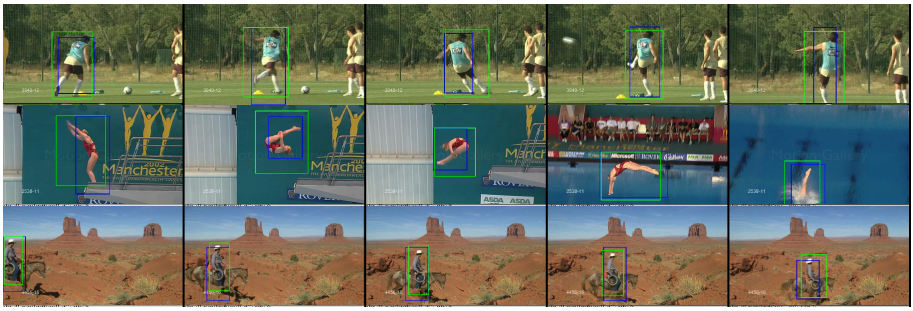

Figure 5.15: Qualitative examples of our action proposals. In each row, we select 5 frames of a video for visualization purpose. Blue boxes are the closest proposal to each ground truth box (shown in green).

coarse-to-fine cascade on multiple layers of a detection network, reusing the features already computed for detection. We have evaluated the method on most recent benchmarks and against previous approaches, and have shown that in most cases it is comparable to or better than state-of-the-art approaches in terms of both accuracy and computation. The source code of DeepProposals is available online[3].

[3]https://github.com/aghodrati/deepproposal

Chapter 6

From Representation to Generation

To address research question 4, in this chapter we define the new problem of generating modified images which aims to generate a face as similar as possible to the input face but with an altered visual attribute. We propose a deep encoder-decoder architecture that learns to generate a face from a representation. Work covered in this chapter is based on:

- Ghodrati A.[1], Jia X.[1], Pedersoli M., Tuytelaars T., *Towards Automatic Image Editing: Learning to See another You*, In Proceedings of the British Machine Vision Conference (BMVC), 2016.

6.1 Introduction

When looking at pictures of family and friends, does any of the following comments sound familiar to you? *"Nice picture, just a pity her eyes are hidden behind those big sunglasses."* or *"I look so old on this picture - wish I could make myself look 5 years younger!"* or *"What would he look like if he grew a beard?"*. Now imagine the next generation of image editing tools, which are able not just to correct red eyes, but also modify specific attributes of the people depicted in photographs: remove their sunglasses, make them look younger

[1]indicates equal contribution: Amir worked more on experimental results while Xu worked more on background overview and implementation. They contributed equally for the rest including main idea and writing.

or add that beard, while keeping all other facial characteristics and imaging conditions constant.

Thoughts along these lines motivated us to start looking into this problem. While our current method cannot yet be applied 'in the wild' in applications like the one sketched above, it does show promising results which make us think that automatic image editing might actually become doable in the near future. What we can do now is shown in Figure 1.4. In this chapter, we work on face images because a)faces have a common structure, b)face datasets are available and rich in terms of number of images, capturing setting and labeling and c)face datasets are widely used in the community.

There have been several works on image generation [A.Dosovitskiy et al., 2015, Denton et al., 2015, Gauthier, 2014, Goodfellow et al., 2014, Hinton et al., 2011, Kingma and Welling, 2014, Kulkarni et al., 2015, Li et al., 2015, Rezende et al., 2014, Tieleman, 2014, Yang et al., 2015, Yim et al., 2015, Zhu et al., 2013, Zhu et al., 2014]. However, note how our problem setting is different from the one tackled by most image generation methods found in the literature. [Denton et al., 2015, Gauthier, 2014, Goodfellow et al., 2014, Li et al., 2015] train a model to generate images from scratch, i.e. without an input image as reference. Others often focus only on changing a face or object pose (e.g. [A.Dosovitskiy et al., 2015, Yang et al., 2015]), or learn to generate faces only for a canonical setting (e.g. looking straight into the camera, with standard diffuse illumination and neutral expression [Yim et al., 2015, Zhu et al., 2013]).

We start with a large dataset of cropped and aligned faces, with a variety of attributes (e.g. pose and illumination) annotated and varied systematically for each individual in the database. We propose a model following the encoder-decoder fashion. It takes a face image as input and encodes it into several feature maps; takes a desired attribute vector as input and encodes it into several feature maps; then combines and deeply fuses these two flows of information; and finally generates a new image with a convolutional decoder module. In addition, in order to generate more realistic images we adopt a coarse-to-fine scheme, dividing the problem in two stages with each one focusing on one aspect. The first stage is in charge of rendering a global representation of the desired object, while the second focuses on local refinements to remove some artifacts. To demonstrate the effectiveness of the proposed method, we evaluate it on the MultiPIE [Gross et al., 2010] dataset for three sub-tasks, that is, changing the face pose, changing the image illumination and also image inpainting.

The qualitative and quantitative results show that the proposed method can generate face images of good quality and keep all information of the input face except what is specified by the desired attribute. We also evaluate the proposed method on a face retrieval task. Given a query image we want to retrieve similar

images but with some altered attributes, similar to what is done in [Ghodrati et al., 2015b]. Instead of learning an attribute detector first, we generate with our method an image with the desired attribute and then take the altered image as query to do standard similarity-based retrieval.

In summary, the main contributions of this chapter are: i) definition of a new problem, where the goal is to generate images as similar as possible to a source image yet with one attribute changed; ii) a solution that follows an encoder-decoder pipeline, where the desired attribute modification is first encoded then integrated at feature map level; iii) the insight that the result can be refined by adding another convolutional encoder-decoder model; and iv) good qualitative and quantitative results on different tasks on the MultiPIE dataset.

The remainder of this chapter is organised as follows. First, in section 6.2, we discuss related work. Section 6.3 describes our proposed method. Section 6.4 details the experimental evaluation of our method, and section 6.6 concludes the chapter.

6.2 Related Work

Recently, there have been several works addressing the task of image generation [A.Dosovitskiy et al., 2015, Denton et al., 2015, Gauthier, 2014, Goodfellow et al., 2014, Hinton et al., 2011, Kingma and Welling, 2014, Kulkarni et al., 2015, Li et al., 2015, Rezende et al., 2014, Tieleman, 2014, Yang et al., 2015, Yim et al., 2015, Zhu et al., 2013, Zhu et al., 2014, Radford et al., 2015, Tatarchenko et al., 2015]. These methods can roughly be divided into two categories.

A first line of works follows an unsupervised approach. Some of these methods learn the distribution of the training images and generate an image from scratch [Denton et al., 2015, Gauthier, 2014, Goodfellow et al., 2014, Li et al., 2015, Radford et al., 2015]. Hinton *et al.* [Hinton et al., 2011] and Tieleman [Tieleman, 2014] proposed the capsule network to disentangle different visual components in the code layer of an autoencoder. Kingma and Welling [Kingma and Welling, 2014] and Rezende *et al.* [Rezende et al., 2014] proposed the variational autoencoder. It applies the re-parameterization trick to the latent variables in the code layer to model different factors of the visual appearance. Kulkarni *et al.* [Kulkarni et al., 2015] proposed to use convolutional variational autoencoder and a special training scheme to learn interpretable graphics code for different appearance transformations.

The second group of works generates images conditioned on either an image or several attributes. Dosovitskiy *et al.* [A.Dosovitskiy et al., 2015] map high-level

information such as chair type and viewpoint into the 2D image space to generate different chair images. Radford *et al.* [Radford et al., 2015] propose guidelines for stable training of deep convolutional Generative Adversarial Networks (GANs). They generate faces conditioned on desired attributes. To this end, they benefit from vector arithmetic properties in the latent space. To have meaningful arithmetic operations, their method relies on smooth learned manifold, however, it is not guaranteed that their method always generates semantically meaningful images as the space is not learned for this task. Gardner *et al.* [Gardner et al., 2015] propose a data-driven method that for every image, it finds a traversal path on the manifold of natural images from input image towards the target image with desired class labels. Zhu *et al.* [Zhu et al., 2013, Zhu et al., 2014], Yim *et al.* [Yim et al., 2015] and Yang *et al.* [Yang et al., 2015] propose to generate images conditioned on an image and an attribute. Given an object in a specific pose, [Zhu et al., 2013] propose a network to generate frontal faces with fixed illumination. Yim *et al.* [Yim et al., 2015] propose to generate a face with desired pose and neutral illumination with a multi-task model which includes a generation DNN and a reconstruction DNN. Yang *et al.* [Yang et al., 2015] use a recurrent network to model pose changes of faces. However they model are specifically for changing the pose and is not designed for changing other attributes. Our method also falls in this category and focuses on the generation of face images. However in contrast to the previously mentioned approaches, our method is not designed specifically for changing the pose of a face; we will show the generality of our network on other tasks. In addition, most methods will change other aspects of a face (*e.g.* illumination or identity) during generation; instead, in our task we want to generate an image with the desired attribute while preserving the other image information as before.

Our work is also related to face image editing such as novel facial image generation. In [Hassner et al., 2015], Hassner *et al.*proposed to use a single 3D surface and symmetry to generate front facing view for all face images. Mohammed *et al.* [Mohammed et al., 2009] proposed a two stage method for novel facial image generation. They first make an estimation using a parametric global model and then use a local non-parametric model to refine the generation. In our paper, we propose a pipeline which consists of an image generation network and an image refinement network and show that it shares the same global-local two stage philosophy.

6.3 Proposed Method

In this section, we describe the proposed convolutional encoder-decoder architecture for image generation. The overview of the proposed method

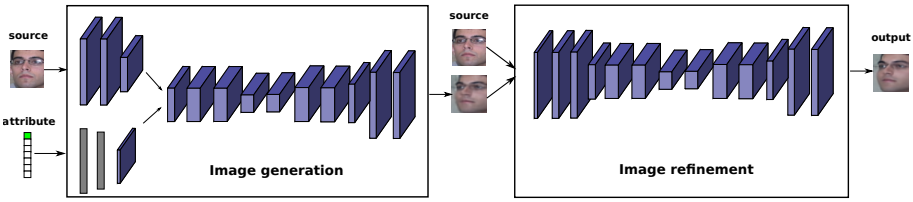

Figure 6.1: An overview of our proposed method. Given a source image and attribute vector, we modify the source based on the attribute vector and generate an image in a two stage approach.

is shown in Figure 6.1. Given a source image and a target attribute vector, our goal is to generate a new image with the target attribute while maintaining as much as possible other aspects of the appearance of the source image. To this end, we propose a two-stage approach, with a first network for image generation (see section 6.3.1) and a second one for image refinement (see section 6.3.2). Let us denote the source image as $\mathbf{X}$ and the target image and the target attribute as $\mathbf{Y}$ and $\mathbf{C_Y}$.

6.3.1 Convolutional encoder-decoder for image generation

Inspired by the recent success of deconvolutional networks in generating accurate images of chairs from high-level descriptions [A.Dosovitskiy et al., 2015] and semantic segmentation [Long et al., 2015, Noh et al., 2015], we adopt a convolutional encoder-decoder architecture for our task. However, in contrast to previous works we deal with two inputs coming from different modalities: an image and a target attribute vector. The architecture can be conceptually divided into four operations (Figure 6.2): image encoding, attribute vector encoding, feature map fusion and image decoding.

Image encoding. In this component, we encode a source image $\mathbf{X}$ into a set of feature maps via two convolutional layers and a max-pooling layer. Inspired by the OxfordNet [Simonyan and Zisserman, 2015] we use two consecutive convolutional layers, each of which includes filters of size (3×3) and one rectifier linear unit (ReLU) [Krizhevsky et al., 2012]. The two consecutive convolutional layers help to increase the receptive field with a reduced increment of the number of parameters. The convolutional layers are able to capture local information of the source image and later we will show how to use them to generate the target image. The structure of this part can be expressed as $\text{Conv}(3, 3, 64)$-

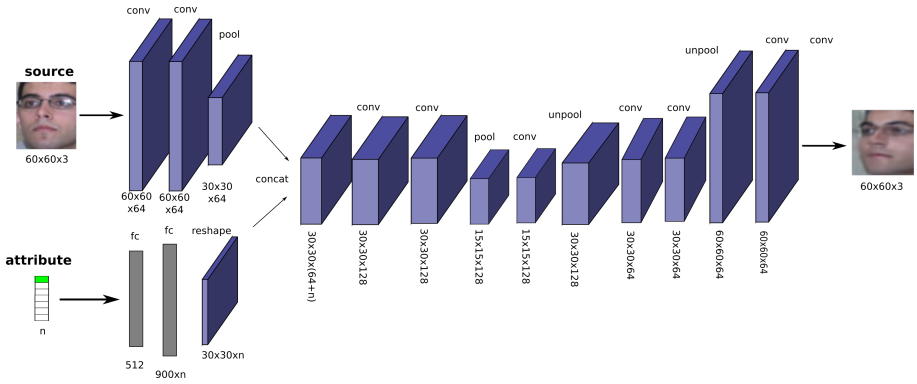

Figure 6.2: The architecture of *image generation* network.

Conv$(3, 3, 64)$-ReLU-Pool$(2, 2)$. Since we use a $60 \times 60 \times 3$ as input, we obtain feature maps of size $30 \times 30 \times 64$ as output of this component.

Attribute vector encoding. We express the attribute as a one-hot vector. The size of this vector depends on the number of attributes used for the corresponding sub-task. For example in the sub-task of changing the pose, since we have 7 distinct poses, the attribute vector has 7 elements with one of them set to 1 (indicating the desired pose) and the rest to 0. In order to convert the knowledge contained in the attribute vector into visual information we convert it into feature maps which can be easily combined with the convolutional features of the input image. We apply two fully connected layers to the one-hot vector and then reshape it to feature maps of size $30 \times 30 \times n_a$, where n_a is the number of values for an attribute. The structure of this part can be expressed as FC(512)-FC$(900 \times n_a)$-Reshape$(30, 30, n_a)$.

Feature map fusion. In this step, we fuse the feature maps obtained from the encoding of the input image and the attribute vector. The feature maps with attribute information can propagate their message to the feature maps extracted from the source input. The two sets of feature maps are first stacked together, generating a new set of feature maps of $30 \times 30 \times (64 + n_a)$. Then a feature map fusion layer similar to the cross channel pooling layer in [Lin et al., 2014a, Szegedy et al., 2014] is applied on top to fuse the two sets of feature maps. The cross channel pooling layer can be viewed as a $(1, 1)$ convolutional layer followed by a ReLU. In our case, the feature map fusion layer is more sophisticated, including several consecutive convolutional layers. The convolution operations compute the weighted average of feature maps

so that it allows sufficient interaction between feature maps of the source image and the attributes in a learning framework. The structure of this part can be expressed as Concat-Conv$(3, 3, 128)$-ReLU-Conv$(3, 3, 128)$-ReLU-Pool$(2, 2)$-Conv$(1, 1, 128)$-ReLU. The output of this step is a feature map of size $15 \times 15 \times 128$.

Image decoding. Once the feature maps computed from source image $\mathbf{X}$ and the attribute vector $\mathbf{C_Y}$ have been fully integrated, the next step is to generate an image with the same size as the input image. This module aggregates local information from different feature maps. Similar to [A.Dosovitskiy et al., 2015], a deconvolution module here consists of a 2×2 unpooling layer and two convolutional layers. The unpooling layer doubles the size of feature maps in the previous layer by replacing each element of the feature map with a 2×2 block. The top left corner is filled in with the element of the feature map and the rest are filled with zeros. The unpooling layer is followed by two consecutive $(3, 3)$ convolutional layers. Note that all convolutional layers have a ReLU activation except the last one because the output should have both positive and negative values after input images being normalized to zero mean and one standard deviation. The structure of this part can be expressed as Unpool$(2, 2)$-Conv$(3, 3, 64)$-ReLU-Conv$(3, 3, 64)$-ReLU-Unpool$(2, 2)$-Conv$(3, 3, 64)$-ReLU-Conv$(3, 3, 3)$. The output of this step is an image of size $60 \times 60 \times 3$.

6.3.2 Image generation refinement

The ideal generated image should have good quality, have the desired target attribute and keep the appearance of the input image. It is difficult for a single network to generate an image that satisfies all the above requirements simultaneously since as the architecture becomes deeper, the risk of overfitting goes higher. Therefore, we adopt the divide-and-conquer scheme, dividing the problem into two stages. The output of the proposed convolutional encoder-decoder network produces already a reasonable result, but it still has some missing details and some artifacts. As shown in Figure 6.3, we propose to add another convolutional encoder-decoder network to perform image refinement in the second stage. The second stage takes as input the source image and the generated image of the first stage.

These two inputs are first concatenated channel-wise. Then we apply several convolutional, ReLU and max-pooling layers in the encoding process followed by unpooling, convolutional and ReLU layers in the decoding process. Convolutional layers locally fuse the information from two inputs and refine the output of the first stage network.

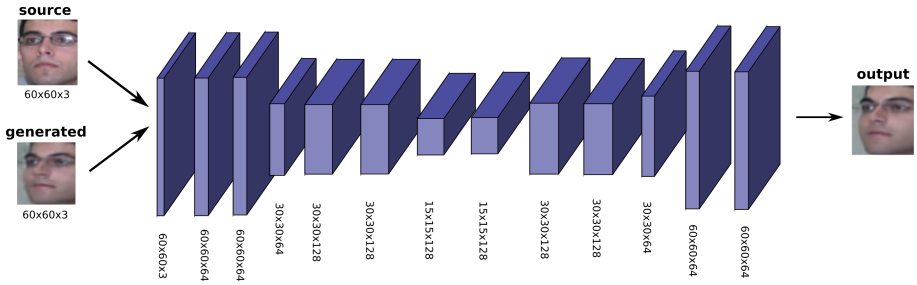

Figure 6.3: The architecture of *iamge refinement* network.

Also, note that for decoding the image, we use different set of parameters than image encoding. The architecture of the whole network is as follows: Concat - Conv$(3, 3, 64)$ - ReLU - Conv$(3, 3, 64)$ - ReLU - Pool$(2, 2)$ - Conv$(3, 3, 128)$ - ReLU - Conv$(3, 3, 128)$ - Pool$(2, 2)$ - Conv$(1, 1, 128)$ - ReLU-Unpool$(2, 2)$ - Conv $(3, 3, 128)$ - ReLU - Conv$(3, 3, 64)$ - ReLU - Unpool$(2, 2)$ - Conv$(3, 3, 64)$ - Conv$(3, 3, 3)$.

Our method shares similar philosophy as the coarse-to-fine two stage method used for synthesizing novel faces [Mohammed et al., 2009]. In [Mohammed et al., 2009], they first use a global parametric model to make an approximate estimation of the global structure of a face. Then a local non-parametric model is conditioned on the global estimation to refine the initial result of the global model. In our framework, the goal of the first-stage network is to generate an image that has the correct global structure of the target face. In the second stage, the goal is to keep the visual consistency with the outputs of the first stage but removing artifacts and adding the details to the face. We show the effectiveness of such paradigm for image generation in section 6.4.

6.3.3 Training

We train the two networks successively. The input of the first network are the source image of size $60 \times 60 \times 3$ and a one-hot attribute vector. Also, in each network the encoder and decoder have their own, separate parameters and are trained together. Each image is preprocessed by subtracting the mean and dividing by the standard deviation. Then, the output of the first network and the source image are fed as input to the second network. For both networks,

we use Mean Squared Error (MSE) as loss function.

$$L_1(W_1) = ||F_1(X, C_Y; W_1) - Y||_2^2, \tag{6.1}$$

$$L_2(W_2) = ||F_2(X, \hat{Y_1}; W_2) - Y||_2^2, \tag{6.2}$$

Where X, C_Y and $\hat{Y_1}$ are input image, desired attribute vector and the output of first stage respectively. W_i is the model parameters for the network i. The training is carried out using mini-batch gradient descent with backpropagation [LeCun et al., 1989] and the batch size is set to 32. We use a fixed momentum of 0.95 and learning rate of $1E-6$. All the weights are initialized with the method proposed in [He et al., 2015a].

6.4 Experiments

In order to demonstrate the effectiveness of the proposed method we evaluate it for three different tasks. The main task is to rotate the face and is carried out on the MultiPIE dataset [Gross et al., 2010]. We extensively evaluate our method for this task, showing both qualitative and quantitative results. The other two tasks are generating faces with different illumination on the MultiPIE dataset and filling in a missing part for face images generated from MultiPIE. We show some quantitative and qualitative results for these two tasks as well.

6.4.1 Rotating faces

The session 1 of the MultiPIE dataset consists of images of 249 identities under 15 poses and 20 different illumination conditions. We select a subset of it that covers 7 poses ($-45°$ to $+45°$). The first 100 subjects are used for training and the rest are used for testing. All faces are aligned and cropped based on eyes and chin annotations provided by Shafey *et al.* [Shafey et al., 2013], then resized to 60×60 pixels.

Here we consider pose as an attribute and aim at generating faces with the same identity and illumination as the input image but with the desired pose. The input to our method is an image and a target pose vector. To this end, during training we build a set of image pairs, where the images show the same person, with the same illumination, but with different pose (i.e. $100 \times 20 \times P_7^2 = 84000$ pairs). The first element of a pair is considered as input image of the network and the second element is the ground-truth target image.

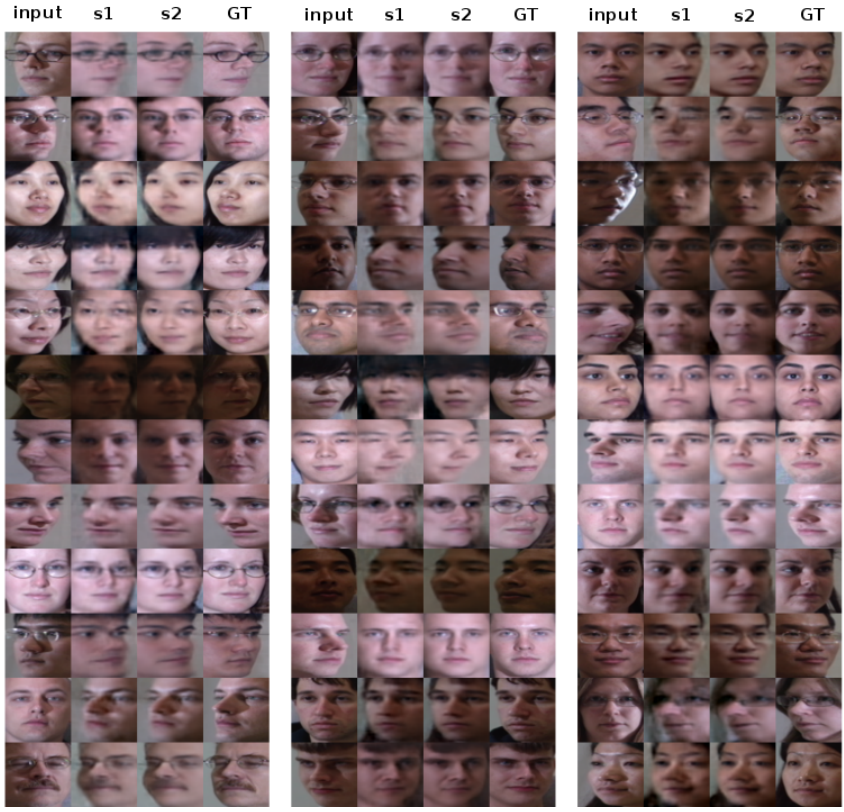

Figure 6.4: Some qualitative results of our image generation from test data of MultiPIE. In each row, the first column is the input image, the last column is the ground-truth target image, the 2nd column is the output of first stage and the 3rd column is generated image of second stage network.

Image Generation Results. In Figure 6.4 we show some qualitative results, i.e., the generated images from both stages of the method. We can see that the proposed method can generate face images that are visually similar to the target face. Notice how the generated faces from the second stage have better image quality and details than the first stage output.

We qualitatively compare our method with CPI [Yim et al., 2015] in Figure 6.5. From the figure, we can see that our method (left) preserves the source image information better than theirs (right), that is, more details of the faces and less clutter and noise. However, note that the two approaches have been trained

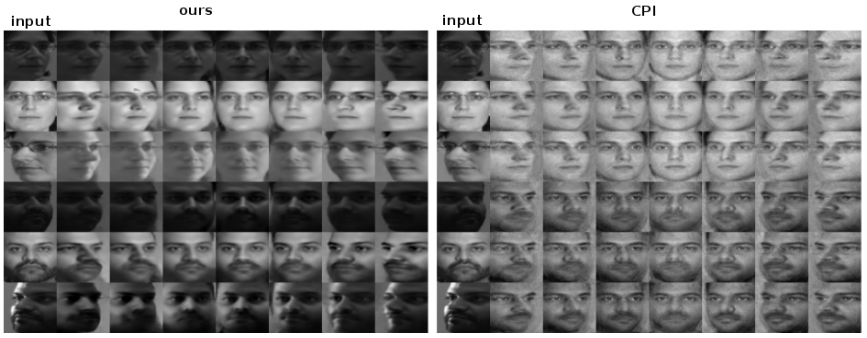

Figure 6.5: Qualitative comparison between our generated images (**Left**) and [Yim et al., 2015] (**Right**). On each set the first column is the input face and next 7 ones are generated faces in different poses. We convert our generated faces to gray-scale for fair comparison with [Yim et al., 2015].

per-pixel MSE	First stage	Second stage	CPI [Yim et al., 2015]
All	381.5	376.3	–
Neutral illumination subset	578.5	570.5	884.4

Table 6.1: Mean Squared Error on all images and on a subset of 510 images with neutral illumination, for a fair comparison with [Yim et al., 2015].

for different tasks; the network in [Yim et al., 2015] is trained to generate neutral illumination regardless of the input image illumination, while ours aims at keeping the same illumination as in the original image.

It is worth mentioning that [Yim et al., 2015] uses much more complex model than ours. Due to the use of locally linear and fully connected layers, their network has much more parameters than ours (200M vs. 6M parameters). Besides, they add an auxiliary branch for reconstruction in their network to preserve the identity. In contrast, we have a much simpler architecture, which only contains fully connected layers for attribute vector encoding and only convolutional layers elsewhere. The attribute vector encoding module makes a rough estimation about the desired pose. This information is propagated to the later part of the network and deeply fused with the feature maps computed from the input image which contains both identity and illumination information.

We quantitatively validate the effectiveness of our method as well. We randomly select 10000 generated images and compare the performance of each stage in terms of per-pixel mean squared error (MSE) between generation and ground-

truth image and report it in Table 6.1 (first row). The results in the table confirm the visual impression that the second stage generates higher quality images. This implies that the second stage network works as expected: it locally refines the initial generation via additional convolutional layers. We also compare our network with [Yim et al., 2015] in terms of per-pixel MSE. To this end the authors kindly provided us the CPI features (images) that they have used for face recognition in their paper. To have a fair comparison, we only select the subset of test faces with neutral illumination (510 faces) since their network is designed to be illumination-invariant (i.e. it always generates faces with neutral illumination). From Table 6.1 (second row), we observe that our method has a better per-pixel MSE than [Yim et al., 2015], which again confirms the visual impression.

To get some insight about the performance under various pose changes, we compute per-pixel MSE for different pose rotations in Figure 6.6. As expected, small pose changes are easier to deal with than large ones. Besides, we also observe that the task is easier when input and output are symmetric poses (e.g., compare the result from -15 to 15 with the one from -15 to -45) and when the desired pose is closer to frontal face (e.g., compare the MSE from -30 to 0 with the one from 0 to +30).

In addition, in Figure 6.7 we divide the generated test data into 7 groups based on the desired face pose. For each group we compute per-pixel MSE separately. As shown in the figure, the largest pixel errors come from the silhouette of the face, hair and neck regions since there is much uncertainty and it is difficult to generate them properly from a single image.

Retrieval results. To demonstrate how our method preserves other aspects of a face (e.g., identity and illumination) while changing the pose, we conduct an experiment on face retrieval. We generate an image conditioned on a query image and a target pose using the proposed method and then find the K nearest neighbors for the generated face. In our experiment we use L2-normalized features extracted from the 1×1 convolution layer of our second network and measure the distance of query and candidates using Euclidean distance. For the images to retrieve, as we do not know their pose, we pre-compute the features for all possible poses (7) and keep the one whose reconstruction is closer to the original image.

As evaluation criterion, we consider a retrieved image to be correct if its identity and its illumination are the same as the input and its pose is the desired one. The retrieval result for our method is 49.9% and 76.1% for recall@1 and recall@5 respectively. recall@K is defined as the percentage of queries for which the correct image is among the top K retrieved images. Some qualitative results of our retrieval strategy are also shown in Figure 6.8.

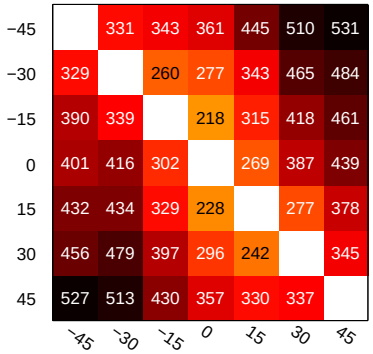

Figure 6.6: Per-pixel MSE for various pose changes. The vertical axis is the source pose rotation and the horizontal is the target pose rotation in degrees.

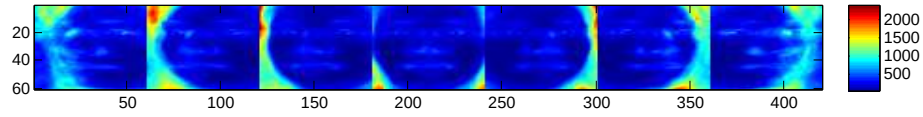

Figure 6.7: Visualization of per-pixel MSE. We grouped the generated faces based on their poses and for each group, we computed per-pixel MSE.

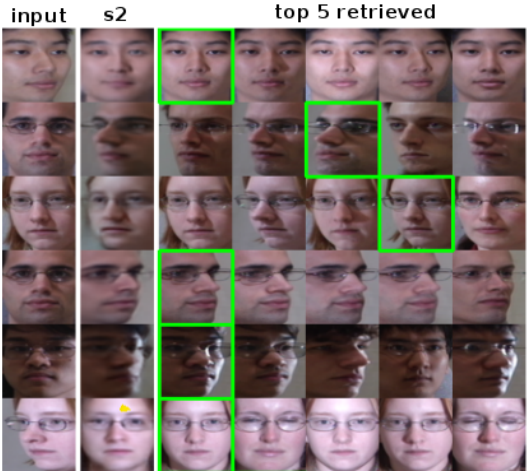

Figure 6.8: Some visual retrieval results. The first column is the query image, the second is the generated image and the remaining columns are the top 5 retrieval results. The green boxes are correct retrieved faces. A retrieved image is considered to be correct if its identity, pose and illumination are correct.

6.4.2 Changing illumination

We train another model to generate a face with desired illumination. To this end, we use the same dataset but change the attribute vector to generate a face with specific illumination out of 20 different illumination conditions. In Figure 6.9, given an input (first column), we generate faces with different illumination by changing the attribute vector to the corresponding desired illumination. Quantitatively, the per-pixel MSE of the test set is 193.3 and 146.6 respectively for the first and second stage. These numbers indicate this task is easier than rotating faces (see Table 6.1). This is expected since it is easier for convolutional filters to learn local illumination changes than propagating appearance information to distant locations. Besides, there is less uncertainty on the silhouette of a face.

6.4.3 Image inpainting

We carry out another interesting experiment on the MultiPIE dataset to demonstrate the ability of the proposed method in image generation. For this task, we randomly generate 10 black blocks of different shapes as occlusion patterns. For each image, one of these 10 patterns is selected and overlaid on the face image at a random location. We train the proposed model to learn to inpaint the occluded face image. In this case, the attribute vector is a binary value which specifies whether a face is occluded or not. As shown in Figure 6.10 our method can generate reasonably good images also for this task considering the high variability of the input image. The model fills in the occluded region using the knowledge that it learned during training such as the continuity of local region and face symmetry. The filled region is visually consistent with the non-occluded parts of the face. Quantitatively, the per-pixel MSE of the test set is 64.6 for the second stage. This number indicates this task is also easier than rotating faces. However, we should notice that in this task, the required modification is not global as in the two other tasks and just a part of the image should be altered. As an evidence, the MSE error between source and target images is just 1473. This explains the small value of the MSE error for this task.

6.5 Discussion

To have a better insight of the method, we test our approach with different kind of attributes that have not been used before. We use images with 7 attributes:

Figure 6.9: Qualitative results for the task of changing illumination. The fist image on the left is the input face. For each identity, the first row shows 20 faces generated under different illuminations and the second row is the corresponding ground-truth

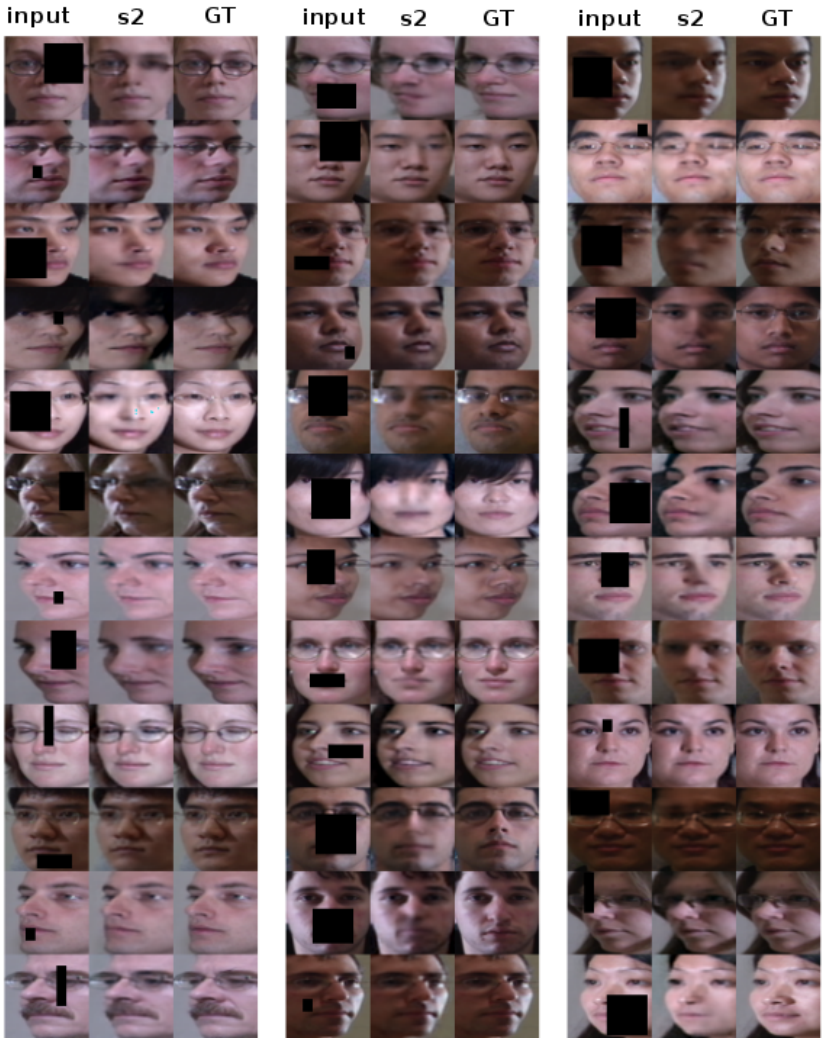

Figure 6.10: Qualitative results for the task of image inpainting. The first column shows the input image, the second column shows images generated with our method and the third column shows the complete image without the occluding pattern.

3 different kind of hats, 3 different kind of glasses and finally a face without any accessory. To this end, we use the CAS-PEAL-R1 [Gao et al., 2008] dataset, separate the data based on person identities and use the first 350 identities for training and the other 88 identities for test. All faces are aligned and cropped based on the eyes and then resized to 60×60 pixels and converted into a grayscale image. In order to train our networks we use the same strategy as for the MultiPIE dataset. We build all possible permutations of pairs of attribute changes (in total 2442).

Figure 6.11 shows some qualitative results of our method for test data. Note that our method can remove one accessory (e.g. hat) and also substitute it with another attribute (e.g. sunglasses) at the same time. As shown in Figure 6.11, also in this case, the second stage of our method (3rd column) refines the generation and creates better quality images compared to our first stage (2nd column).

However, as it is clear, the global quality of the generated images is inferior to the previous task. This is probably due to the fact that different people have different ways of wearing hats and glasses and also people are not under the same condition when wearing different accessories (so the faces are not aligned carefully). Such variations increase the complexity of the data and lead the network to predict an average image of several possible solutions which in practice results in blurred predictions. Finally, the alignment of the faces is only carried out based on the eyes annotation provided in the dataset, and that is far from perfect for some samples. In contrast, in MultiPIE, the images are recorded at the same time with cameras from different views and the alignment is carried out based on both eyes and chin. This explains why for the experiment on MultiPIE dataset we have better alignment and hence better image generation quality than the experiment on CAS-PEAL-R1 dataset.

Figure 6.11: Some qualitative results of our image generation from test data of CAS-PEAL-R1 dataset. Each row contains 2 set of examples. In each example, the first column is the input image, the last column is the desired image, 2nd column is generation of first stage network and 3rd column is the generated image of the second stage network.

6.6 Conclusion

In this work we define a new problem where, given an input image, the goal is to generate images with modified attributes while maintaining as much as possible the similarity to the original image. We propose a solution to this problem, for the case of cropped and aligned faces, in the form of a two stage encoder/decoder convolutional network, with the first stage for image generation and the second one for image refinement. We have validated our approach both qualitatively and quantitatively on the MultiPIE dataset for three sub-tasks. For future work we would like to extend our method to address even more challenging scenarios like dealing with misaligned input faces and applying it to different object categories. More details on future line of work can be found on section 7.3.

Chapter 7

Conclusion & Discussion

In this thesis, we investigated different state-of-the-art image and video representations for multiple challenges in computer vision. These representations allowed the use of learning techniques both for understanding and synthesizing images. During this thesis, we have shown that the way in which images and videos are presented plays an important role in any recognition system. We also showed how a vectorized image representation can be exploited for both recognition and generation. In this chapter we present the conclusions we drew during performing this thesis. As these follow the trends and evolution in the field, we present them in this context. Following, we revisit the research questions that we presented in the introduction. Then we discuss some limitations and provide some directions for future research.

7.1 Conclusion

For more than a decade, bag-of-words approach was the mainstream representation in recognition problems. Bag-of-words is a simple, relatively low-dimensional, geometry-free representation that attempts to reach the final goal of classifying the image, without trying to solve the intermediate problem, that of describing (geometrically) the parts and objects that can appear in an image. Over a decade, researchers proposed many variants of it, improving each of its components. We started the thesis by making use of this representation for the task of action recognition (Chapter 3). Particularly, we showed that if we split the representation of a video into action-related foreground and action-unrelated background and build separate BoW for each of them, we reach to

improved recognition accuracy. Moreover, we concluded from experiments that the quality of the segmentation is an important factor to obtain accurate action recognition.

The next generation of representations, like Fisher encoding, tried to capture more statistics of the data at the cost of having higher dimensionality. In the next chapter, we proposed a pipeline that used solely such 2D representations for estimating the pose. Our results showed how just 2D appearance-based representations enable us to appropriately predict viewpoint of an object and to effectively bypass 3D models. Specifically, we compared three representations namely bag-of-words, Fisher Vector and CNN-based features. We observed that bag-of-words is the poorest method for pose representation. The best representation based on our observations is Fisher Vector with spatial pyramid. CNN-based features perform quite good as well, especially considering their much lower dimensionality. This is somewhat surprising: one would expect the CNN tuned for object recognition to be invariant to viewpoint changes.

After the era of using engineered features and hand-crafted representations, thanks to huge amount of available data and powerful modern hardwares, neural networks, the forgotten child of AI, again came back to the surface. Today, based on the modern deep neural networks, we are able to start fulfilling computer vision prehistoric promises. Considering deep learning as a feature (and representation) learning, it circumvents the challenges of feature extraction: deep learning models are capable of learning informative features and hierarchical representations for making a decision (*e.g.* classification) by themselves, requiring little guidance from the human. This makes deep learning an extremely powerful tool which we have benefited from for generating proposals and detecting objects in our third work. Particularly, we provided an efficient coarse to fine cascade on multiple layers of CNN features that act strongly on object and action locations. Our method generates proposals at low computational cost and we have shown that in most of the cases it is comparable or better than state-of-the-art approaches in terms of both accuracy and computation. In terms of object detection performance, the difference between our CNN-based approach and non-deep approaches is quite significant and it appears mostly in a regime with low number of proposals: our method can outperform non-deep approaches by using almost 6× less amount of proposals.

Observing the power of deep learning, researchers started to tackle more challenging problems that were inaccessible before the deep learning era. One of these tasks is image generation which is very hard due to the high number of parameters that should be estimated. In Chapter 6 we defined a new generation task which aims to generate a face as similar as possible to the input face but with an altered visual attribute (e.g. the same face with a new pose or a different illumination). Our architecture consists of two convolutional networks with the

first stage for image generation and the second one for image refinement. During a set of experiments, we have shown our proposed two-stage, convolutional architecture is able to generate visually appealing faces with a correct modified attribute.

Finally, it is clear I organized the conclusions of my thesis in chronological order. However, as the field of computer vision is progressing rapidly, revisiting the conclusions is needed. As indicating before, the last few years have been marked by exceptional progress that has come from recent advances in deep learning. These models regained their status as a leading paradigm in machine learning. Therefore, recently this idea is applied to many vision tasks including action recognition [Simonyan and Zisserman, 2014, Ji et al., 2013], image segmentation [Long et al., 2015, Noh et al., 2015] and viewpoint estimation [Tulsiani and Malik, 2015]. Particularly, end-to-end training or fine-tuning the deep networks for a specific task led to promising results in most of the cases. So by now, there are state-of-the-art CNN networks that outperform our results in action recognition and pose estimation.

7.2 Revisiting the research questions

At the beginning of this thesis we set the objective of investigating different representations for different computer vision challenges. To this end, we formulated four research questions which were presented in the introduction. From then on, we went on a journey aimed at answering these questions which resulted in a set of contributions. Based on these contributions and observations, we now revisit each of the research questions and address them accordingly.

1. **To what extent does disentangling foreground and background representations affect classification? Is quality of disentangling important for the final task of classification?**

   The results obtained in Chapter 3 suggest that segmentation is crucial for good recognition. Learning a shared segmentation model (called actionness) for segmenting foreground from background helps to have better action classification performance. In addition, we tried to improve the segmentation quality by co-segmentation but the improvement brought by the co-segmentation seems to be limited. Our proposed iterative learning scheme that alternates between segmentation and recognition brought us the best results. However, in iterative learning, linearity is a limitation that should be preserved. An alternative way to obtain similar results is to map the foreground and background representations of the video into a non-linear kernel. Here we observed while for linear models

the best coding is LLC (LLC captures the non-linearity of the data), for non linear kernels hard vector quantization (VQ) is performing better.

2. **Are 2D-based representations enough for estimating viewpoint of an object?**

   The research question was explored by using two modern, 2D-based representations for estimating viewpoint of a given object. In contrast to some trends in the vision community that suggest 3D information about the object class is beneficial for an accurate estimation of the object pose, we used Fisher Vector and CNN-based representations which do not make any assumption or reasoning about the 3D. We found out that a 2D architecture, if properly adapted to the task, can provide top performance. Our results in Chapter 4 confirm that Fisher vectors encoded with spatial information in most of the cases outperform the state-of-the-art, including methods based on 3D or much more complex and computationally expensive models. In addition, almost the same performance can be obtained using deep-learned features.

3. **How information from a hierarchical representation can be exploited for locating objects?**

   Aiming at answering this research question, in Chapter 5 we did a set of experiments and we observed that in a pre-trained CNN, there is not a single best layer for candidate windows generation: Deeper layers, having a more semantic representation, perform very well in recalling the objects but they provide a poor localization of the object. Earlier layers are better in accurately localizing the object of interest, but their recall is reduced as they do not represent strong object cues. Leveraging the hierarchical nature of deep architectures, we proposed a method based on a cascade starting from the last convolutional layer that has a coarse spatial window resolution, going down with subsequent spatial refinements until the initial layers of the net. Our proposed cascade achieved higher object recall even with a small number of windows.

4. **To what extent is a representation inversely convertible to an image?**

   To answer this research question, we defined the new problem of generating modified images in Chapter 6. In general, image generation is a very challenging task in computer vision. This is due to the high number of variables that need to be estimated in image space. Moreover, even with low regression error, generating an image that is visually appealing for human vision is another challenge. To find out to what extent a representation is convertible to the image space, we proposed a deep encoder-decoder that learns to generate a convertible representation. Our

proposed encoder learns to map the input image and the desired change into a set of feature maps. Then, after fusing both representations, a decoder is trained to convert the representation to an image. During several experiments on faces, we saw that the proposed architecture can generate face images that are visually similar to the target face. We observed that this model generates faces with reasonable details and limited noise. We also quantitatively validated the generated faces by computing per-pixel mean squared error (MSE) between generated and ground-truth face. The numbers confirm the visual impression of the generated images. However, since this task is hard, we had to impose some limitation on the input data. For example, our input image resolution is 60×60 and also they are cropped and aligned carefully.

7.3 Discussion and Directions for future research

As discussed before, a lot of progress in computer vision has come from recent advances in deep learning. However, in video classification, unlike image recognition, these deep learning based methods fail to outperform previous hand-crafted features. Most of current deep learning based action recognition methods equally treat temporal and spatial domain while there is intrinsic differences between them *i.e.* we can not consider a video as a 3D image. Proposing methods that capture long and short term dynamics of the videos is an interesting, challenging problem that helps action recognition to go further.

For estimating viewpoint, we proposed a pipeline in Chapter 4 which sequentially does object detection and viewpoint estimation. Emerging CNN-based detectors give us two opportunities for having an improved performance: i) they are proved to produce promising detection results and ii) their representation can implicitly capture spatial relationships among the different object parts which may be informative for viewpoint estimation. Therefore, the intuitive way to tackle this problem is to do both of these tasks simultaneously using the rich information CNN provides. One possibility is to design a unified end-to-end CNN framework that directly predicts object bounding box and effectively estimates its viewpoint using the multi-task learning paradigm. In addition, landmark localization can be achieved just by stacking a few layers thanks to the convolution architecture. Another limitation of our proposed method is that currently it just handles discrete angles while naturally viewpoint is a continuous parameter. By combining powerful 2D representations with 3D reasoning we may have the best of both worlds.

In the task of object proposal generation and object detection, as we concluded

in Chapter 5, our top-down reasoning combined with a low-level refinement led to top performance in proposal generation. On the other hand, bottom-up reasoning has the advantage of localizing the object more accurately. This is usually thanks to the segmentation map they build during proposal generation. One direction for the future of object proposal generation is to take the best of both worlds in a unified framework where a network can satisfy both high recall and also high overlap constraints. Also, it is shown that false positives in the proposal stage negatively affect the detection performance. Another direction for next generation object proposals is to propose methods that produce fewer proposals while maintaining high recall. This may be obtained by a tighter integration between the proposal generator and the detector.

Our work described in Chapter 6 is one of the pioneers towards automatic image editing. Even though at this moment our method cannot yet be applied 'in the wild', it does show promising results which make us think that this task might actually become doable in the near future. The convolutional encoder-decoder that we proposed is trained to minimize the pixel-wise reconstruction error of the generated image compared to the ground-truth image. This error term is problematic for images since translation is punished disproportionately to the small error perceived by human vision. As a consequence, cropping and aligning carefully the faces before feeding it to the network is crucial otherwise the algorithm is biased towards generating smooth images with a correct global subject. Recently, Generative Adversarial Networks (GANs) are proposed by [Goodfellow et al., 2014] which use another type of loss function to generate images. The loss in GAN models is calculated by another discriminative network which tries to distinguish between generated and real images. The goal of these networks is generating an image as similar as possible to the real images such that the classification error in a discriminative network be maximised. Here, 'real' means that the image came from our training set of images in contrast to the generated fakes. Such design is in favor of generating images with more correct local style and less emphasizing on the global structure. We believe there are open areas to investigate the ways of combining these two fundamentally different methods which may lead to interesting results.

Finally, looking at the horizon of computer representation, we can observe that recent deep learning based representations merely solve pattern recognition problems, *i.e.* learning is the process of discovering patterns in an image through multi layers of feature learning. While they are powerful for prediction, such representations are poor in terms of understanding and explaining the world. Current models perceive the world without causality which in many cases leads to revealing errors (see Figure 7.1). We believe that richer, more structured representations can be obtained by incorporating relational representations into current models. The aim for statistical model relations is that if an interpretation

does not satisfy a causal relation, it becomes less probable, but not necessarily impossible. Even though, modeling dependencies between examples can be much more complex than treating examples independently, such structured representations are more similar in terms of form to representations in human brains and may be able to causally reason about the world.

Figure 7.1: Describing scenes without causality: Image captions generated by a deep neural network [Karpathy and Fei-Fei, 2015]. The Figure is borrowed from [Lake et al., 2016].

Bibliography

[A.Dosovitskiy et al., 2015] A.Dosovitskiy, J.T.Springenberg, and T.Brox (2015). Learning to generate chairs with convolutional neural networks. In *CVPR*. pages 96, 97, 99, 101

[Alexe et al., 2010] Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *CVPR*. pages 26, 32, 64, 66, 68, 81, 87

[Arbelaez et al., 2011] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *PAMI*. pages 79

[Arbelaez et al., 2014] Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *CVPR*. pages 64, 67, 87

[Arie-Nachimson and Basri, 2009] Arie-Nachimson, M. and Basri, R. (2009). Constructing implicit 3d shape models for pose estimation. In *ICCV*. pages 47

[Berg et al., 2014] Berg, T., Liu, J., Lee, S. W., Alexander, M., Jacobs, D., and Belhumeur, P. (2014). Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*. pages 51

[Bergh et al., 2013] Bergh, M., Roig, G., Boix, X., Manen, S., and Gool, L. (2013). Online video seeds for temporal window objectness. In *ICCV*. pages 68

[Bilen et al., 2011] Bilen, H., Namboodiri, V., and Van Gool, L. (2011). Object and action classification with latent variables. In *BMVC*. pages 26

[Bilen et al., 2015] Bilen, H., Pedersoli, M., and Tuytelaars, T. (2015). Weakly supervised object detection with convex clustering. In *CVPR*. pages 64

[Bowan and Optometrist, 2012] Bowan, M. D. and Optometrist, O. N. (2012). Integrating vision with the other senses. pages 2

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. In *PAMI*. pages 33

[Brendel and Todorovic, 2010] Brendel, W. and Todorovic, S. (2010). Activities as time series of human postures. In *ECCV*. pages 42

[Brox and Malik, 2010] Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*. pages 25, 29, 30, 92

[Cai et al., 2014] Cai, Z., Wang, L., Peng, X., and Qiao, Y. (2014). Multi-view super vector for action recognition. In *CVPR*. pages 43

[Carreira et al., 2012] Carreira, J., Caseiro, R., Batista, J., and Sminchisescu, C. (2012). Semantic segmentation with second-order pooling. In *ECCV*. pages 49

[Carreira and Sminchisescu, 2012] Carreira, J. and Sminchisescu, C. (2012). Cpmc: Automatic object segmentation using constrained parametric min-cuts. *PAMI*. pages 67

[Cheng et al., 2014] Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. (2014). Bing: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*. pages 64, 66, 67, 71, 81, 87

[Cinbis et al., 2013] Cinbis, R. G., Verbeek, J., and Schmid, C. (2013). Segmentation driven object detection with fisher vectors. In *ICCV*. pages 64

[Crammer and Singer, 2001] Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292. pages 21

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*. pages 50, 52

[Denton et al., 2015] Denton, E. L., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*. pages 96, 97

[Deselaers et al., 2010] Deselaers, T., Alexe, B., and Ferrari, V. (2010). Localizing objects while learning their appearance. In *ECCV*. pages 27, 64

[Dollár and Zitnick, 2013] Dollár, P. and Zitnick, C. L. (2013). Structured forests for fast edge detection. In *ICCV*. pages 67, 71, 74

[Donahue et al., 2013] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*. pages 19, 46, 50

[Endres and Hoiem, 2010] Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *ECCV*. pages 26, 32

[Erhan et al., 2014] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *CVPR*. pages 67

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*. pages 5, 52, 69, 76, 81

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*. pages 73

[Fanelli et al., 2011] Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *CVPR*. pages 47

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. In *PAMI*. pages 34, 48

[Fulkerson et al., 2009] Fulkerson, B., Vedaldi, A., Soatto, S., et al. (2009). Class segmentation and object localization with superpixel neighborhoods. In *ICCV*. pages 26

[Gaidon et al., 2011] Gaidon, A., Harchaoui, Z., and Schmid, C. (2011). A time series kernel for action recognition. In *BMVC*. pages 36, 41, 42

[Gaidon et al., 2012] Gaidon, A., Harchaoui, Z., and Schmid, C. (2012). Recognizing activities with cluster-trees of tracklets. In *BMVC*. pages 25, 27, 28

[Gao et al., 2008] Gao, W., Cao, B., Shan, S., Chen, X., Zhou, D., Zhang, X., and Zhao, D. (2008). The CAS-PEAL large-scale chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(1):149–161. pages 111

[Gardner et al., 2015] Gardner, J. R., Kusner, M. J., Li, Y., Upchurch, P., Weinberger, K. Q., and Hopcroft, J. E. (2015). Deep manifold traversal: Changing labels with convolutional features. *arXiv preprint arXiv:1511.06421*. pages 98

[Gauthier, 2014] Gauthier, J. (2014). Conditional generative adversarial nets for convolutional face generation. Class Project forStanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014. pages 96, 97

[Ghodrati et al., 2015a] Ghodrati, A., Diba, A., Pedersoli, M., Tuytelaars, T., and Van Gool, L. (2015a). Deepproposal: Hunting objects by cascading deep convolutional layers. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 66

[Ghodrati et al., 2015b] Ghodrati, A., Jia, X., Pedersoli, M., and Tuytelaars, T. (2015b). Swap retrieval: Retrieving images of cats when the query shows a dog. In *ICMR*. pages 97

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *ICCV*. pages 66, 76, 87

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*. pages 67, 83, 87

[Girshick et al., 2010] Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. (2010). Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/. pages 48, 54

[Gkioxari and Malik, 2015] Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *CVPR*. pages 69, 75, 92

[Glasner et al., 2011] Glasner, D., Galun, M., Alpert, S., Basri, R., and Shakhnarovich, G. (2011). Viewpoint-aware object detection and pose estimation. In *ICCV*. pages 47, 58, 61

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*. pages 96, 97, 118

[Gross et al., 2010] Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2010). Multi-pie. *Image and Vision Computing*, 28(5). pages 52, 96, 103

[Grundmann et al., 2010] Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *CVPR*. pages 29

[Gu and Ren, 2010] Gu, C. and Ren, X. (2010). Discriminative mixture-of-templates for viewpoint classification. In *ECCV*. pages 46, 47

[Hariharan et al., 2014] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Hypercolumns for object segmentation and fine-grained localization. In *arXiv preprint arXiv:1411:5752*. pages 64

[Hasan et al., 2016] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., and Davis, L. S. (2016). Learning temporal regularity in video sequences. *arXiv preprint arXiv:1604.04574*. pages 20

[Hassner et al., 2015] Hassner, T., Harel, S., Paz, E., and Enbar, R. (2015). Effective face frontalization in unconstrained images. In *CVPR*. pages 98

[He et al., 2015a] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*. pages 103

[He et al., 2015b] He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Spatial pyramid pooling in deep convolutional networks for visual recognition. *PAMI*. pages 64, 83, 87

[Hejrati and Ramanan, 2012] Hejrati, M. and Ramanan, D. (2012). Analyzing 3d objects in cluttered images. In *NIPS*. pages 46, 47

[Hinton et al., 2011] Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In *ICANN*. pages 96, 97

[Hoai et al., 2011] Hoai, M., Lan, Z., and De la Torre, F. (2011). Joint segmentation and classification of human actions in video. In *CVPR*. pages 26

[Hochbaum and Singh, 2009] Hochbaum, D. S. and Singh, V. (2009). An efficient algorithm for co-segmentation. In *ICCV*. pages 26, 32

[Hoiem and Huang, 2015] Hoiem, D. and Huang, J.-B. (2015). University of illinois computer vision course cs543/ece549. `http://www.slideshare.net/jbhuang/lecture-21-image-categorization-visionspring2015`. pages 13

[Hosang et al., 2015] Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2015). What makes for effective detection proposals? *PAMI*. pages 77, 79, 81, 87

[Jaakkola and Haussler, 1999] Jaakkola, T. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. *NIPS*. pages 49

[Jain et al., 2014a] Jain, A., Tompson, J., Andriluka, M., Taylor, G. W., and Bregler, C. (2014a). Learning human pose estimation features with convolutional networks. In *CVPR*. pages 48

[Jain et al., 2014b] Jain, M., Gemert, J., Jégou, H., Bouthemy, P., and Snoek, C. (2014b). Action localization with tubelets from motion. In *CVPR*. pages 68, 92

[Jain et al., 2013] Jain, M., Jegou, H., and Bouthemy, P. (2013). Better exploiting motion for better action recognition. In *CVPR*. pages 43

[Ji et al., 2013] Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *PAMI*. pages 115

[Karpathy and Fei-Fei, 2015] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR*. pages 119

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*. pages 44

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *ICLR*. pages 96, 97

[Kovashka and Grauman, 2010] Kovashka, A. and Grauman, K. (2010). Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *CVPR*. pages 42

[Krähenbühl and Koltun, 2014] Krähenbühl, P. and Koltun, V. (2014). Geodesic object proposals. In *ECCV*. pages 67

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*. pages 18, 46, 50, 64, 77, 99

[Kulkarni et al., 2015] Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. (2015). Deep convolutional inverse graphics network. In *NIPS*. pages 96, 97

[Lake et al., 2016] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2016). Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*. pages 119

[Lan et al., 2011] Lan, T., Wang, Y., and Mori, G. (2011). Discriminative figure-centric models for joint action localization and recognition. In *ICCV*. pages 27, 36, 42, 91

[Laptev and Lindeberg, 2003] Laptev, I. and Lindeberg, T. (2003). Space-time interest points. In *ICCV*. pages 24

[Laptev et al., 2008] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *CVPR*. pages 24

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*. pages 15, 50, 73

[LeCun et al., 1989] LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551. pages 103

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. pages xiii, 15, 17, 18

[Lezama et al., 2011] Lezama, J., Alahari, K., Sivic, J., and Laptev, I. (2011). Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*. pages 25, 30

[Li et al., 2015] Li, Y., Swersky, K., and Zemel, R. S. (2015). Generative moment matching networks. In *ICML*. pages 96, 97

[Liebelt and Schmid, 2012] Liebelt, J. and Schmid, C. (2012). Multi-view object class detection with a 3d geometric model. In *CVPR*. pages 47, 61

[Lin et al., 2014a] Lin, M., Chen, Q., and Yan, S. (2014a). Network in network. In *ICLR*. pages 100

[Lin et al., 2014b] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014b). Microsoft coco: Common objects in context. In *ECCV*. pages 81, 88

[Liu et al., 2009] Liu, J., Luo, J., and Shah, M. (2009). Recognizing realistic actions from videos "in the wild". In *CVPR*. pages 36

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., and Reed, S. (2016). Ssd: Single shot multibox detector. In *ECCV*. pages 67

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*. pages 99, 115

[Lopez-Sastre et al., 2011] Lopez-Sastre, R. J., Tuytelaars, T., and Savarese, S. (2011). Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshops*. pages 46, 47, 52, 54, 58, 61

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*. pages 12, 49

[Manen et al., 2013] Manen, S., Guillaumin, M., and Gool, L. V. (2013). Prime object proposals with randomized prim's algorithm. In *ICCV*. pages 67, 68, 81, 87

[Masci et al., 2011] Masci, J., Meier, U., Ciresan, D. C., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *ICANN*. pages 19

[Mohammed et al., 2009] Mohammed, U., Prince, S. J. D., and Kautz, J. (2009). Visio-lization: generating novel facial images. *ACM Trans. Graph.* pages 98, 102

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *ICCV*. pages 99, 115

[Oneata et al., 2014] Oneata, D., Revaud, J., Verbeek, J., and Schmid, C. (2014). Spatio-temporal object detection proposals. In *ECCV*. pages 68, 92

[Oneata et al., 2013] Oneata, D., Verbeek, J., and Schmid, C. (2013). Action and event recognition with fisher vectors on a compact feature set. In *ICCV*. pages 43

[Ozuysal et al., 2009] Ozuysal, M., Lepetit, V., and Fua, P. (2009). Pose estimation for category specific multiview object localization. In *CVPR*. pages 47, 52, 58

[Payet and Todorovic, 2011] Payet, N. and Todorovic, S. (2011). From contours to 3d object detection and pose estimation. In *ICCV*. pages 61

[Pepik et al., 2012a] Pepik, B., Gehler, P., Stark, M., and Schiele, B. (2012a). 3d2pm–3d deformable part models. In *ECCV*. pages 47, 58, 61

[Pepik et al., 2012b] Pepik, B., Stark, M., Gehler, P., and Schiele, B. (2012b). Teaching 3d geometry to deformable part models. In *CVPR*. pages 46, 47, 59, 60, 61

[Perronnin et al., 2010] Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *ECCV*. pages 14, 49

[Pinheiro et al., 2015] Pinheiro, P. O., Collobert, R., and Dollar, P. (2015). Learning to segment object candidates. In *NIPS*. pages 68

[Prest et al., 2012] Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *CVPR*. pages 27

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. pages 97, 98

[Raptis et al., 2012] Raptis, M., Kokkinos, I., and Soatto, S. (2012). Discovering discriminative action parts from mid-level video representations. In *CVPR*. pages 25, 27, 30, 41, 42

[Razavian et al., 2014] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*. pages 46

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *CVPR*. pages 68

[Redondo-Cabrera et al., 2014] Redondo-Cabrera, C., Lopez-Sastre, R., and Tuytelaars, T. (2014). All together now: Simultaneous object detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. In *BMVC*. pages 47

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*. pages 64, 68, 87, 88

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*. pages 96, 97

[Rodriguez et al., 2008] Rodriguez, M. D., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*. pages 36, 90, 91

[Rosenfeld and Weinshall, 2011] Rosenfeld, A. and Weinshall, D. (2011). Extracting foreground masks towards object recognition. In *ICCV*. pages 26

[Rubio et al., 2012] Rubio, J. C., Serrat, J., and López, A. (2012). Video co-segmentation. In *ACCV*. pages 27

[Sapienza et al., 2012] Sapienza, M., Cuzzolin, F., and Torr, P. (2012). Learning discriminative space-time actions from weakly labelled videos. In *BMVC*. pages 27, 28, 42

[Savarese and Li, 2007] Savarese, S. and Li, F.-F. (2007). 3d generic object categorization, localization and pose estimation. In *ICCV*. pages 52, 54

[Sermanet et al., 2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. pages 68

[Shafey et al., 2013] Shafey, L. E., McCool, C., Wallace, R., and Marcel, S. (2013). A scalable formulation of probabilistic linear discriminant analysis: Applied to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1788–1794. pages 103

[Shapovalova et al., 2012] Shapovalova, N., Vahdat, A., Cannons, K., Lan, T., and Mori, G. (2012). Similarity constrained latent support vector machine: An application to weakly supervised action classification. In *ECCV*. pages 27, 42

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. In *PAMI*. pages 30

[Simonyan et al., 2013] Simonyan, K., Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2013). Fisher vector faces in the wild. In *BMVC*. pages 47

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *NIPS*. pages 44, 115

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*. pages 19, 81, 99

[Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *ICCV*. pages 12

[Song et al., 2014] Song, H. O., Girshick, R., Jegelka, S., Mairal, J., Harchaoui, Z., and Darrell, T. (2014). On learning to localize objects with minimal supervision. In *ICML*. pages 64

[Soomro et al., 2012] Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*. pages 75, 91

[Stalder et al., 2012] Stalder, S., Grabner, H., and Van Gool, L. (2012). Dynamic objectness for adaptive tracking. In *ACCV*. pages 32

[Stark et al., 2010] Stark, M., Goesele, M., and Schiele, B. (2010). Back to the future: Learning shape models from 3d cad data. In *BMVC*. pages 47

[Su et al., 2015] Su, H., Qi, C. R., Li, Y., and Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*. pages 61

[Su et al., 2009] Su, H., Sun, M., Fei-Fei, L., and Savarese, S. (2009). Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*. pages 47

[Sun et al., 2009] Sun, M., Su, H., Savarese, S., and Fei-Fei, L. (2009). A multi-view probabilistic model for 3d object classes. In *CVPR*. pages 47

[Szegedy et al., 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. In *CVPR*. pages 100

[Tatarchenko et al., 2015] Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2015). Single-view to multi-view: Reconstructing unseen views with a convolutional network. *arXiv preprint arXiv:1511.06702*. pages 97

[Tieleman, 2014] Tieleman, T. (2014). *Optimizing neural networks that generate images*. PhD thesis, University of Toronto. pages 96, 97

[Todorovic, 2012] Todorovic, S. (2012). Human activities as stochastic kronecker graphs. In *ECCV*. pages 42

[Toshev and Szegedy, 2014] Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *CVPR*. pages 47

[Tran et al., 2014] Tran, D., Yuan, J., and Forsyth, D. (2014). Video event detection: From subvolume localization to spatiotemporal path search. *PAMI*. pages 69

[Tulsiani and Malik, 2015] Tulsiani, S. and Malik, J. (2015). Viewpoints and keypoints. In *CVPR*. pages 61, 115

[Ullah et al., 2010] Ullah, M. M., Parizi, S. N., and Laptev, I. (2010). Improving bag-of-features action recognition with non-local cues. In *BMVC*. pages 26, 35

[Van de Sande et al., 2011] Van de Sande, K. E., Uijlings, J. R., Gevers, T., and Smeulders, A. W. (2011). Segmentation as selective search for object recognition. In *ICCV*. pages 64, 67, 68, 77, 87

[van Gemert et al., 2015] van Gemert, J. C., Jain, M., Gati, E., and Snoek, C. G. (2015). Apt: Action localization proposals from dense trajectories. In *BMVC*. pages 69, 90, 92

[Vezhnevets et al., 2011] Vezhnevets, A., Ferrari, V., and Buhmann, J. M. (2011). Weakly supervised semantic segmentation with a multi-image model. In *ICCV*. pages 27, 32

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *IJCV*. pages 65

[Wang et al., 2011] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2011). Action recognition by dense trajectories. In *CVPR*. pages 25, 28, 35, 36, 42

[Wang et al., 2009] Wang, H., Ullah, M. M., Kläser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. In *BMVC*. pages 24

[Wang et al., 2010] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *CVPR*. pages 13, 14, 29

[Wang et al., 2015] Wang, L., Qiao, Y., and Tang, X. (2015). Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314. pages 44

[Wang et al., 2013] Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013). Regionlets for generic object detection. In *ICCV*. pages 64, 67

[Weinzaepfel et al., 2015] Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). Learning to track for spatio-temporal action localization. In *ICCV*. pages 69

[Xiang et al., 2014] Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*. pages 52, 54, 59, 60

[Xinggang et al., 2015] Xinggang, W., Yan, W., Xiang, B., and Zhijiang, Z. (2015). Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*. pages 71

[Xu and J.Corso, 2012] Xu, C. and J.Corso, J. (2012). Evaluation of super-voxel methods for early video processing. In *CVPR*. pages 29

[Yang et al., 2015] Yang, J., Reed, S., Yang, M.-H., and Lee, H. (2015). Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*. pages 96, 97, 98

[Yim et al., 2015] Yim, J., Jung, H., Yoo, B., Choi, C., Park, D., and Kim, J. (2015). Rotating your face using multi-task deep neural network. In *CVPR*. pages xv, 96, 97, 98, 104, 105, 106

[Yu and Yuan, 2015] Yu, G. and Yuan, J. (2015). Fast action proposals for human action detection and search. In *CVPR*. pages 69

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*. pages 83

[Zhao et al., 2014] Zhao, Q., Liu, Z., and Yin, B. (2014). Cracking bing and beyond. In *BMVC*. pages 67

[Zhou et al., 2014] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *NIPS*. pages 79

[Zhu and Ramanan, 2012] Zhu, X. and Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *CVPR*. pages 46, 47, 52, 58, 59

[Zhu et al., 2013] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2013). Deep learning identity-preserving face space. In *ECCV*. pages 96, 97, 98

[Zhu et al., 2014] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2014). Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*. pages 96, 97, 98

[Zia et al., 2013] Zia, M. Z., Stark, M., Schiele, B., and Schindler, K. (2013). Detailed 3d representations for object recognition and modeling. *PAMI*, 35(11). pages 47

[Zitnick and Dollár, 2014] Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *ECCV*. pages 64, 66, 67, 71, 74, 87

Curriculum

Amir Ghodrati was born on 25^{th} December 1984, in Shahrood, Iran. In the nationwide university entrance exam held in 2003, he was ranked 240^{th} among $450,000$ applicants. He received the degree of Computer Engineering with a major in software engineering from AmirKabir University of Technology (Tehran Polytechnic) in Iran. In 2007 he was ranked 10^{th} among $10,000$ applicants of nationwide graduate entrance exam in Artificial Intelligence. He received the degree of Master of Science in Artificial Intelligence from the Sharif University of Technology in Tehran with GPA of 17.91/20. During his master studies, he worked on Human Action Recognition Using Spatio-Temporal Local Features. Amir joined the industry just after graduation where he served as a designer and developer of License Plate Recognition and Speedometer systems in Tehran. In December, 2011, he joined the PSI computer vision lab at KU Leuven (Belgium) as a predoc under supervision of Prof. dr. Tinne Tuytelaars. Next, in March, 2013, he qualified as a PhD candidate in the same lab under the advise of Prof. dr. Tinne Tuytelaars. During his PhD he researched on various challenging computer vision tasks and published several papers in leading computer vision conferences like ICCV, CVPR, ECCV and BMVC.

List of publications

Journal Articles

- Fernando B., Gavves E., Oramas M.J., Ghodrati A., Tuytelaars T. (2016) *Rank Pooling for Action Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence
- Ghodrati A., Diba A., Pedersoli M., Tuytelaars T., Van Gool L. *DeepProposals: Hunting Objects and Actions by Cascading Deep Convolutional Layers. Submitted to* International Journal of Computer Vision.

Conference Articles

- Ghodrati A., Pedersoli M., Tuytelaars T. (2014). *Coupling video segmentation and action recognition* . Proceedings of the IEEE Winter Conference on Applications of Computer Vision - (WACV). Steamboat Springs, CO, USA, 24-26 March 2014.
- Ghodrati A., Pedersoli M., Tuytelaars T. (2014). *Is 2D Information Enough For Viewpoint Estimation?.* Proceedings of the British Machine Vision Conference - (BMVC). Nottingham, UK, 1-5 September 2014.
- Fernando B., Gavves E., Oramas M.J., Ghodrati A., Tuytelaars T. (2015). *Modeling video evolution for action recognition.* Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition - (CVPR). Boston, MA, USA, 7-10 June 2015.
- Ghodrati A., Jia X., Pedersoli M., Tuytelaars T. (2015). *Swap Retrieval: Retrieving Images of Cats When the Query Shows a Dog.* Proceedings of the 5th ACM on International Conference on Multimedia Retrieval - (ICMR) Shanghai, China, June 23-26, 2015.

- Ghodrati A., Diba A., Pedersoli M., Tuytelaars T., Van Gool L. (2015). *DeepProposal: Hunting Objects by Cascading Deep Convolutional Layers.* Proceedings of the IEEE International Conference on Computer Vision - (ICCV). Santiago, Chile, 13-16 December 2015.
- Ghodrati A., Jia X., Pedersoli M., Tuytelaars T. (2016). *Towards Automatic Image Editing: Learning to See another You.* Proceedings of the British Machine Vision Conference - (BMVC). York, UK, 19-22 September 2016
- De Geest R., Gavves E., Ghodrati A., Li Z., Snoek C., Tuytelaars T. (2016). *Online Action Detection.* Proceedings of European Conference on Computer Vision – (ECCV). Amsterdam, Netherlands, 11-14 October 2016.

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING
PSI
Kasteelpark Arenberg 10 box 2441
B-3001 Heverlee
amir.ghodrati@esat.kuleuven.be
http://homes.esat.kuleuven.be/∼aghodrat